Contents

- Download TVS
  - Using the TVS Launcher
  - As a Static website
  - As a Docker image
- Changelog
  - 5.11.0 (2026-05-10)
  - 5.10.0 (2026-04-20)
  - 5.9.2 (2026-03-16)
  - 5.9.1 (2026-03-15)
  - 5.9.0 (2026-03-09)
  - 5.8.0 (2026-02-09)
  - 5.7.3 (2026-01-17)
  - 5.7.2 (2026-01-15)
  - 5.7.1 (2026-01-10)
  - 5.7.0 (2025-12-28)
  - 5.6.0 (2025-12-20)
  - 5.5.0 (2025-12-14)
  - 5.4.1 (2025-11-23)
  - 5.4.0 (2025-10-25)
  - 5.3.1 (2025-09-18)
  - 5.3.0 (2025-09-14)
  - 5.2.1 (2025-09-07)
  - 5.2.0 (2025-08-18)
  - 5.1.1 (2025-07-31)
  - 5.1.0 (2025-04-30)
  - 5.0.0 (2025-03-30)
  - 4.5.0 (2025-01-27)
  - 4.4.0 (2025-01-12)
  - 4.3.0 (2025-01-05)
  - 4.2.2 (2024-12-22)
  - 4.2.1 (2024-12-21)

- 4.2.0 (2024-12-19)
- 4.1.0 (2024-12-02)
- 4.0.1 (2024-11-23)
- 4.0.0 (2024-11-21)
- 3.5.0 (2024-10-17)
- 3.4.0 (2024-09-25)
- 3.3.0 (2024-09-21)
- 3.2.1 (2024-09-15)
- 3.2.0 (2024-09-09)
- 3.1.0 (2024-09-07)
- 3.0.0 (2024-08-31)
- 2.7.1 (2024-08-29)
- 2.7.0 (2024-08-28)
- 2.6.1 (2024-08-24)
- 2.6.0 (2024-08-20)
- 2.5.0 (2024-08-11)
- 2.4.0 (2024-07-27)
- 2.3.0 (2024-07-08)
- 2.2.0 (2024-07-07)
- 2.1.0 (2024-07-02)
- 2.0.0-beta (2024-06-20)
- 2.0.0-alpha6 (2024-05-16)
- 2.0.0-alpha5 (2024-05-13)
- 2.0.0-alpha4 (2024-05-12)
- 2.0.0-alpha3 (2024-05-11)
- 2.0.0-alpha2 (2024-05-10)
- 2.0.0-alpha1 (2024-05-09)
- 1.0.0 (2024-03-31)
- 0.6.2 (2023-02-06)
- 0.6.1 (2018-09-30)
- 0.6.0 (2017-05-03)
- 0.5.3 (2017-05-01)

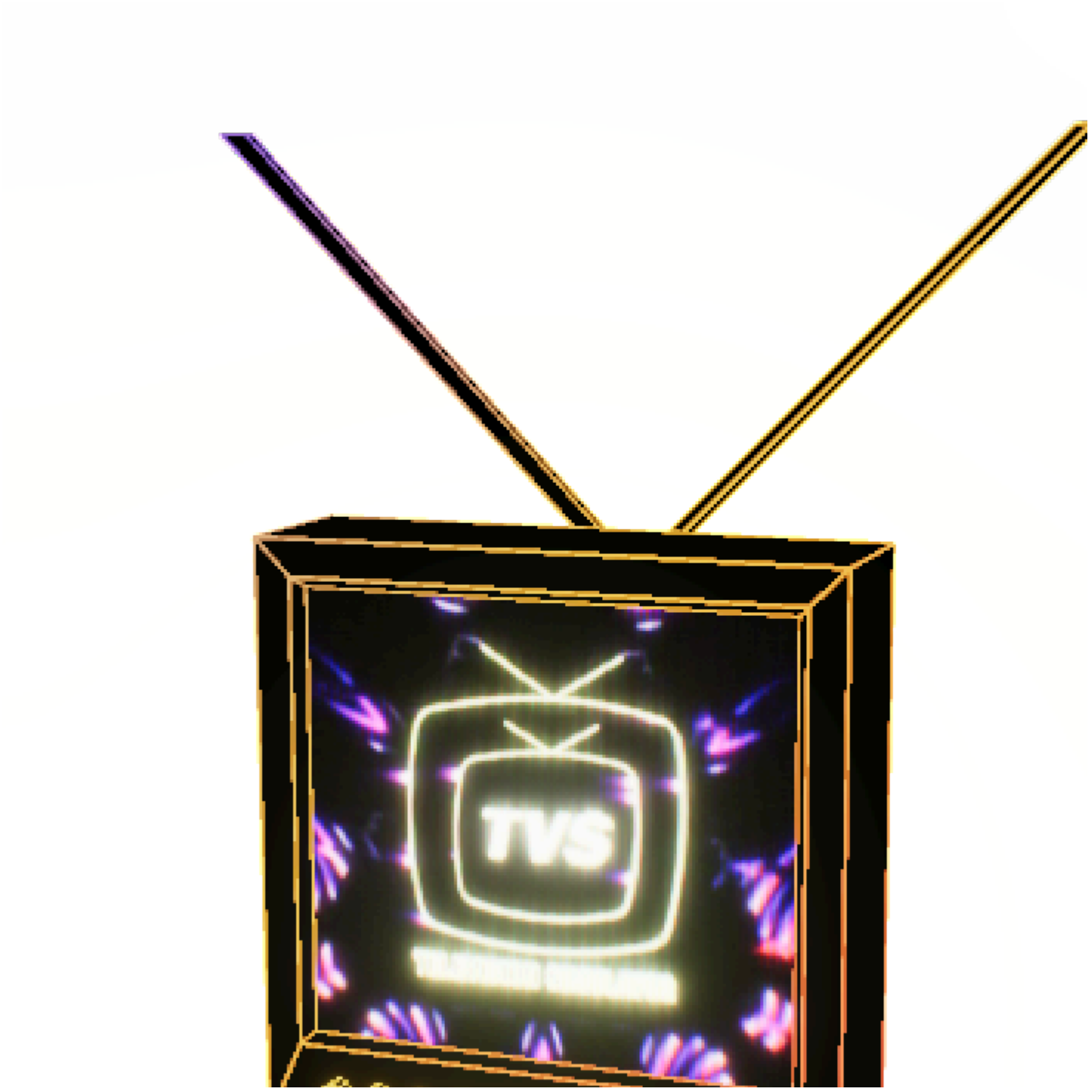
    - 0.4.0 (2017-05-01)
    - 0.3.3 (2017-04-29)
    - 0.3.2 (2017-04-29)
    - 0.3.1 (2017-04-29)
    - 0.0.1-preview (2017-04-28)
- Get Started
    - Compatibility
    - Using a Web Server
    - Using Docker
- Configuring TVS
    - Conceptual overview
    - What’s a YML file?
    - Reading the example config
    - Channels and content tutorial
- Playing Background Audio
    - Radio streams
    - Using a Playlist
- Running as a Kiosk
    - On Linux (Raspberry Pi OS, etc.)
    - On Windows
    - CORS Mitigation
- Using the Remote
    - Web Remote
    - Physical Remote
- Background Audio Options
    - Properties
- Channels
    - Example
    - Channel properties
- Compatibility
    - Browser Configuration
- Keyboard Shortcuts
    - Overriding Keyboard Shortcuts
- Remote Configuration

    - Properties
- Content Engine Primer
    - Examples
    - Properties
    - _component section
    - interruptsBackgroundAudio
    - Shorthand Properties
- Abstract Background
    - Properties
- Character Generator
    - Using the Bulletin Editor
    - Using the Character Generator Engine
    - Properties
- Color
    - Properties
- Daily Schedule
    - Wrapping around midnight
    - Properties
- Generator
    - Test Pattern
    - Color Bars
    - Grid
    - Properties
- IFrame
- Images
    - Properties
- Keno
    - Properties
- Loop
    - Properties
- Refs (Referenced Content)
    - refs section
    - Overriding properties
    - Nested references

- Slideshow
  - Properties
- Video
  - Properties
- Visualizer
  - Properties
- Weather
  - Properties
- Youtube Playlist
  - Properties
- Layout Primer
  - ”A” and “B” components
- Horizontal Split
  - Properties
- Vertical Split
  - Properties
- Mix
  - Properties
  - Layer Properties
- Picture-in-Picture
  - Properties
- Launcher Changelog
  - 1.1.0 (2026-05-24)
  - 1.0.0-preview2 (2025-11-23)
  - 1.0.0-preview (2025-11-21)
- Using the Launcher
  - Installation
  - Usage
  - Using the CLI
- Developer Portal
- TVS IFrame API
  - Handshake Process
  - Options Format
  - Example Usage

    - Integration with TV State
    - Security Considerations
    - Test Application
- Credits, Thanks, Acknowledgements
    - 3rd Party Software
    - 3rd Party Music and Artwork
    - Special Thanks
- Support the Project!
    - Supporter Benefits

Download TVS



Using the TVS Launcher

Download the TVS Launcher for your platform, move it to its own folder and run it. The launcher will automatically download and update TVS for you. See Using the Launcher for more information.

Download Launcher for Windows

Download Launcher for macOS

Download Launcher for Raspberry Pi

Download Launcher for other Linux distributions on GitHub

Next Steps

Using the Launcher →

As a Static website

You can also download the latest static website version of TVS and host it yourself. This is useful for shared web hosting, GitHub Pages or other static site hosts or if you have a web server already set up.

Download on GitHub

Next Steps

Getting Started with a Static Website →

As a Docker image

You can run TVS as a Docker container. This is useful for self-hosting or deploying to cloud platforms that support Docker.

View on Docker Hub

```
docker run -d \
  --name tvs-server \
  -p 80:3000 \
  -v /path/to/your/config.tvs.yml:/home/static/config.tvs.yml \
  -v /path/to/your/content:/home/static/content \
  --restart unless-stopped \
  zshall/television-simulator:latest
```

docker-compose.yml

```
services:
  tvs-server:
    image: 'zshall/television-simulator:latest'
    restart: unless-stopped
    ports:
      - '80:3000'
    volumes:
      - /var/tvs/config.tvs.yml:/home/static/config.tvs.yml
      - /var/tvs/content:/home/static/content
```

Next Steps

Getting Started with Docker →

Changelog

5.11.0 (2026-05-10)

New Features

Added a new virtual video signal feature that works with all components. Think of it as the ability to define a resolution that all inner components will use. Added a new _component property, present on all content engines, that allows you to define properties that can apply to any component. Currently we only use one property: `dimensions` (the virtual video signal).

```
- number: 270
  name: Virtual Video Signals
  abbr: VVS
  generator:
    type: test-pattern
    _component:
      dimensions:
        width: 1280
        height: 720
        aspectRatioBehavior: contain
```

Bug Fixes

- Fixed a bug with the validation schema where `volume` was incorrectly set as a required property for the `generator` engine.
- Fixed validation schema bug where `loop` didn't correctly inherit from the base component.

Deprecation Notice

In version 6.0.0, `interruptsBackgroundAudio` will change; either being moved to `_component` or restructured in another way. That way all properties that apply to all components will be under `_component`.

5.10.0 (2026-04-20)

New Features

This update has added a new Character Generator engine that can be used in many scenarios:

- Displaying news bulletins, weather and text ads
- Overlaying text over videos or images
- Adding a crawl / ticker to a channel
- Integrating with live data (given you supply it in the right format)

TVS 5.10 ships with a built-in character generator template editor which you can also try on the demo site.

This update also adds two new weather providers that work worldwide: wttr.in and weatherapi.com which requires an API key.

```
- number: 150
  name: Weather demo
  abbr: WX1
  weather:
    logo: placeholders/weather.png
    provider: wttr.in
    units: imperial

- number: 151
  name: Weather demo
  abbr: WX2
  weather:
    logo: placeholders/weather.png
    provider: weatherapi.com
    apiKey: "YOUR-API-KEY"
    units: metric
```

Deprecation Notice

The `basic-text` engine has been deprecated and will be removed in TVS 6.0. All example usages of it have been updated to use `character-generator` instead.

5.9.2 (2026-03-16)

Bug Fixes

- Fixed #115 which was a problem I traced back to a Svelte framework issue. Svelte 5.53.7 exhibited this issue but Svelte 5.53.12 has fixed it. The issue manifested itself in TVS as a problem when changing channels too quickly; we would unload a channel rapidly while it's still loading and this would cause multiple channels to be loaded at the same time despite using a `{#key}` block to prevent this. The zombie channels would never unload and would stick around until we refreshed the page.

- Added a 200ms throttle on channel change operations to help prevent any future issues with changing channels too fast. If you hold down the channel up / down button we no longer change channels as fast as possible; we'll change channels about 5 channels per second instead. If you mash channel up and down buttons rapidly we'll ignore any button presses shorter than 200ms apart.

- Fixed an issue with the YouTube playlist engine where if you use it inside a `h-split` or `v-split` layout the player would take up 100% of the **screen** rather than 100% of its **container** meaning that we were losing half the image. YouTube playlists resize to fit their container properly now.

5.9.1 (2026-03-15)

Bug Fixes

- Fixed #113; the `<marquee>` tag is broken in Chrome 146, so I switched to a custom implementation. This runs at 60 FPS and behaves similarly to the real marquee tag. Currently it is only available in the `1989` program guide theme, but it'll be used in upcoming engines as well.

5.9.0 (2026-03-09)

New Features

- Added new `sat1992` OSD theme based on C-Band satellite TV from the early 90's. This theme brings with it some interesting customization options; true to the way

the real receivers work, we simulate a satellite dish on a motorized base. You can optionally define a list of known satellites with positions based on motor pulses (these values can be any arbitrary number). You can then define `themeOptions` in individual channels to specify the satellite that the picture would be received from. When you go to a channel that uses a different satellite than the one that you were on previously, the simulated motor will display progress saying it’s turning to the position that you specified.

```
+osd:
+  theme: sat1992 # 1990's analog C-band satellite receiver theme
+  themeOptions:
+    # showChannelInterference: false # by default, we display overlays of
+    satellite: C1 # the default satellite for all channels to be on. If unset,
 only channels with a satellite option set will display one.
+    satellites: # you can define a list of different satellites; they'll be a
 letter and number combination followed by a 4-digit number representing the
 location in the sky
+      C1: 3600 # these are all arbitrary numbers
+      C3: 4100
+      C4: 4300
+      C5: 4550
+      G1: 4550
+      G2: 4700
+      G3: 5100
+      G4: 6200
+      T3: 6900

 channels:
 - number: 1
   name: Test Channel
   color: blue
+  themeOptions: # optional section for theme-specific metadata
+    satellite: C3 # you can designate a channel as being on a particular
 satellite; choose from the list you defined in the OSD themes

 - number: 2
   name: Test Channel 2
   color: red
+  themeOptions:
+    satellite: G2

 - number: 3
   name: Test Channel 3
   color: green # if a channel doesn't specify a satellite, it defaults to the
 one specified in the OSD options, or no satellite if one isn't specified there
 either
```

- Added a new `abstract-background` engine, which can currently apply two different visual effects to an image of your choice; `wave1` and `wave2`.

```
 - number: 143
   name: Abstract background 1
+  abstract-background:
+    type: wave1

 - number: 144
   name: Abstract background 2
+  abstract-background:
+    type: wave2

 - number: 145
   name: Abstract background with custom image
+  abstract-background:
+    src: placeholders/example-program.webp
+    type: wave2
+    width: 400
+    height: 300
```

5.8.0 (2026-02-09)

New Features

- Added new `daily-schedule` engine that allows you to specify content to show at different times of the day. Acts as a timed loop (using your browser's time zone).
- Added `pixelated` option to the `image` engine that allows you to use pixelated scaling when the image is scaled up, which can be desirable for low-resolution images or pixel art.

```
- number: 240
  name: Daily Schedule
  abbr: SCHED
  daily-schedule: # The daily schedule engine allows you to define content based on a 24-hour schedule. You can go overnight, but you can't overlap content. The schedule will repeat each day.
    noSignalContent: # Optional. You can specify content to show when there's nothing else scheduled.
      color: black

    content:
    - startTime: "20:00" # Time to start this item, in 24 hour format. Required.
      endTime: "05:00" # Time to end this item, in 24 hour format. Required.
      image: # The content to show. Required.
        src: placeholders/schedule-night.png
        pixelated: true

    - startTime: "05:00"
      endTime: "11:00"
      image:
        src: placeholders/schedule-morning.png
        pixelated: true

    - startTime: "11:00"
      endTime: "20:00"
      image:
        src: placeholders/schedule-day.png
        pixelated: true
```

Bug Fixes

- Fixed an issue where content inside loops couldn’t access channel information.

5.7.3 (2026-01-17)

Bug Fixes

- Fixed an issue with Docker deployments where the example config file wasn’t included in the image. The documentation about getting started with Docker with an example config file should work now.

5.7.2 (2026-01-15)

Bug Fixes

- Fixed a longstanding issue where the Web Remote QR code doesn’t display the Peer ID correctly.
- Fixed issue where the Web Remote QR code would display on localhost or 127.0.0.1, which can’t be reached from other devices. The pairing code is still shown in these cases.
- Changed Web Remote pairing code to be alphanumeric to further prevent collisions.

5.7.1 (2026-01-10)

Bug Fixes

- Added a 10-second cooldown when `backgroundAudio` fails to load before retrying to prevent potentially thousands of requests in a short period of time if the audio source isn’t available.

Deprecation Notice

- Television Simulator 6.0.0 will replace the external listing feature of the program guide with a new system designed to work with the currently-in-development Television Simulator Studio companion server. The `guide.externalListings` option will be removed in 6.0.0 and in its place will be a new API to sync guide data from an external source. More details will be provided closer to release. The `title` and `description` options for channels will also be deprecated in favor of a new `info` option that will include more advanced functionality; more information will be provided later.

⚙ Example Config File Changes

- Added TVS Update Channel (feed reader of the latest changelog updates) on channel 4.
- Changed the “no content” channel to channel 8.

5.7.0 (2025-12-28)

New Features

- Added new TV state messages to the IFrame API: `TVS.Volume`, `TVS.Muting` and `TVS.Power`. These messages inform IFrame applications about changes in volume, mute and power states, respectively. See the TVS IFrame API documentation for more details on how to use these new messages.

- Added `background` option to the PiP layout that allows you to set a background color behind the PiP content.

- Added `aspectRatioBehavior` option to the Video engine that allows you to control how videos are displayed in their container. Options are `contain`, `cover` and `stretch`. Default is `contain`.

```
- number: 2
  name: You're Watching Television Simulator
  abbr: TVS
  video:
    src: placeholders/tvs-loop.mp4
    aspectRatioBehavior: cover # Optional. The aspect ratio behavior of the
video. Values are contain, cover, or stretch. Defaults to contain.
```

5.6.0 (2025-12-20)

New Features

- Added the beginnings of a new IFrame API that allows web applications embedded in an IFrame engine to access options defined in the TVS config file.

See the TVS IFrame API documentation for more details on how to use this feature.

⚙ Other Changes

- Added new placeholder video loop: `placeholders/tvs-loop.mp4`, a seamless 4:3 loop of the TVS logo.
- Updated the example config file to include the placeholder loop on channel 2. Channel 3 is the new `WELCOME` channel.

5.5.0 (2025-12-14)

New Features

- Added a new `limit` option to the loop engine that allows for more advanced content selection capabilities.

Consider this example:

```yaml
refs:
  loop-1:
    loop:
      shuffle: random
      limit: 1
      content:
      - image: placeholders/sequence/1.jpg
        duration: 3
      - image: placeholders/sequence/2.jpg
        duration: 3

  loop-2:
    loop:
      shuffle: random
      limit: 2
      content:
      - image: placeholders/sequence/3.jpg
        duration: 3
      - image: placeholders/sequence/4.jpg
        duration: 3
      - image: placeholders/sequence/5.jpg
        duration: 3

channels:
- number: 185
  loop:
  - ref: loop-1
  - ref: loop-2
```

We'll loop from `loop-1` to `loop-2` then back again, but each time we start up one of these, we'll choose only one item from `loop-1` and only two items from `loop-2`. The nested loops and shuffling makes it work!

Bug Fixes

- Fixed an issue where parents of nested loops (loops within loops) would advance to the next content after the default duration (30 seconds) instead of waiting for an inner loop to finish.

5.4.1 (2025-11-23)

Bug Fixes

- Picture static (noise) loads more efficiently by caching noise frames and reusing them, reducing CPU usage and improving animation smoothness. We render at 60fps instead of as fast as possible.
- Fixed #103 where `noise: 0` wasn't actually disabling noise. For global noise, `noise: 0` will set the default noise to 0 per channel but can still be overridden per-channel, while `noise: false` will globally disable noise no matter what.
- Basic Text engine now supports custom aspect ratios properly.
- Video engine better supports custom aspect ratios as we removed a hard-coded reference to 4:3 aspect ratio.

5.4.0 (2025-10-25)

New Features

- Added Picture-In-Picture (PiP) layout. You can use it to overlay one piece of content on top of another in a smaller window. You can specify the position and size of the PiP window, including a border color and thickness.

```
- number: 132
  name: PIP Layout
  pip:
    a:
      image: placeholders/example-program.webp
    b:
      generator: test-pattern
```

```yaml
- number: 134
  name: Lots of Layouts
  pip:
    borderColor: yellowgreen
    borderSize: 8
    bottom: 40
    right: 20
    a:
      h-split:
        gapSize: 8
        gapColor: white
        splitPercentage: 0.75
        a:
          v-split:
            splitPercentage: 0.75
            gapSize: 8
            gapColor: white
            a:
              image: placeholders/example-ad.webp
            b:
              color: red
        b:
          color: yellow
    b:
      generator:
        type: test-pattern
        volume: false
```

- Added configurable split percentages (0.0 -> 1.0) for HSplit and VSplit layouts.

Bug Fixes

- Custom aspect ratios are supported for Mix layout.
- Fixed a bug in calculating layer sizes regarding quoted numbers.
- Abstract backgrounds cover their available container size.
- Updated `web.config` for IIS deployments to set correct MIME types for WebP files.

⚙ Other Changes

- Updated demo config file with examples of `animateChannelChange` and `changeChannelNoise`.

5.3.1 (2025-09-18)

Bug Fixes

- Allowed `iframe` to be used with custom aspect ratios. They'll now fill the screen properly instead of being forced to 4:3.

5.3.0 (2025-09-14)

New Features

- Added startup video feature. You can now specify content to show when the TV is powered on before the regular channels are shown. This can be a video, image, or any other content type. You can specify a duration for how long to show it or have it display until you interact with the TV (by changing the channel). You can have it display whenever the page loads or whenever the power state is changed (pressing `P` on the keyboard).

```
startup:
  content:
    video: placeholders/hypercom-boot.mp4
```

```
startup:
  content:
    color: blue
  duration: 5
  displayOnPageLoad: true
  displayOnPowerOn: false
```

Optimizations

- Some placeholder images have been converted to `WebP` format to reduce file size. If you are updating an existing installation and you're using placeholder

content, you need to either keep the old `GIF` files or update your config file to use the `WebP` versions instead. Affected files are:

    - `example-ad.gif` -> `example-ad.webp`
    - `example-program.gif` -> `example-program.webp`
- Removed the full-size welcome video since it's quite large and wasn't ever used in the default config file.
- Compressed all background images to further reduce file size.

⚙ Other Changes

- Added documentation links in help, setup and about screens.

5.2.1 (2025-09-07)

Bug Fixes

- Fixed an issue where the YouTube player would not properly seek to the correct video index when resuming playback. #61
- Improved error handling for YouTube API requests.

Troubleshooting Improvements

- Pressing `alt + d` will now log the current channel properties to the console.

5.2.0 (2025-08-18)

I'm publishing the changelog on the docs website now.

New Features

- Added new visualizer options, including customizable presets, duration and blend duration.

```
- number: 85
  abbr: PL1
  name: Nightwave Plaza
  description: Nightwave Plaza radio 24/7
  visualizer:
    src: http://radio.plaza.one/mp3 # If you want to focus entirely on music you can use the visualizer engine which uses the Milkdrop engine from Winamp
    visualization: 27 # Optional. Visualization to start with. Leave empty or set to 0 for a random one or choose any number from 1 to 100 to use that preset.
    visualizationDuration: 20 # Optional. How many seconds before choosing a new random visualization. Defaults to 30. Use 0 to stay on the same visualization indefinitely.
    visualizationBlendDuration: 1  # Optional. How many seconds to blend visualizations together when switching. Defaults to 2.
```

Known Issues

It appears that Twitch streams may not auto-play anymore after recent updates to Twitch's player. I'm not sure about the future of Twitch integration given this update, but to workaround it I've enabled user interactivity in the player so that you can click the play button to start the stream. Users are reporting errors in Chromium browser but not in Firefox at this time. Due to the nature of how TVS works a permanent fix may be outside of my control.

5.1.1 (2025-07-31)

Join our new Discord server!

Bug Fixes

- Fixes #83: YouTube Playlist Engine is broken
- Fixes #81: `blur: false` in config file doesn't disable OSD blur

5.1.0 (2025-04-30)

New features:

- Twitch engine added! You can now embed Twitch.tv streams using the channel's name. The Twitch player will respond to TVS volume controls. You need to be using a secure context to embed Twitch (per their rules) so running on localhost or on HTTPS with a domain name will work.

```
- number: 210
  abbr: TWITCH
  name: Twitch Stream
  twitch: twitchsports # You can embed Twitch streams using the channel's name
```

- Added support for dynamic references. If you define a `refs` section, you can define references to content and use it across multiple channels. You can even include references to other references as long as it's not a circular reference that'd cause an infinite loop.

```yaml
refs: # Optional. You can define content here that can be reused in multiple places. This is useful for things like an ad that you want to show on multiple channels.
  ad1: # This is a reference to an ad that can be used in multiple places.
    image:
      src: placeholders/example-ad.gif # The image to show for the ad.

  intro: # This is a reference to an intro that can be used in multiple places.
    video: placeholders/tvs-intro-sd.m4v # The video to show for the intro.

  ad2:
    mix:
      a:
        ref: ad1 # You can use references inside other references; but be careful of circular references!
        mixMode: difference
      b:
        ref: intro
        mixMode: exclusion

channels:
- number: 190
  name: Referenced Content
  abbr: REF
  ref: ad1 # You can reference content defined in the refs section of the config file. This is useful for reusing content in multiple places.

- number: 191
  name: Loops with References
  abbr: REF2
  loop:
    content:
    - ref: ad1 # You can also use references in loops.
      duration: 3
    - ref: intro
    - image: placeholders/sequence/0.jpg
      duration: 3
    - image: placeholders/sequence/1.jpg
      duration: 3
    - image: placeholders/sequence/2.jpg
      duration: 3
```

```yaml
- number: 192
  name: Layouts with References
  abbr: REF3
  h-split:
    a:
      ref:
        name: ad1 # You can also use references in mixes.
        overrides: # You can override the properties of the referenced content here. IMPORTANT: this only works if the referenced content doesn't use shorthand properties.
          # So for instance if you use `image: xyz.png` this won't work. You need to use `image: { src: xyz.png }` instead to do the override.
          aspectRatioBehavior: contain # Consult the properties of the referenced content for what you can override here.
    b:
      ref: intro

- number: 193
  name: References with overrides
  abbr: REF4
  ref:
    name: ad1 # You can also use references in mixes.
    overrides: # You can override the properties of the referenced content here. IMPORTANT: this only works if the referenced content doesn't use shorthand properties.
    # So for instance if you use `image: xyz.png` this won't work. You need to use `image: { src: xyz.png }` instead to do the override.
      aspectRatioBehavior: contain # Consult the properties of the referenced content for what you can override here.

- number: 194
  name: References with other references
  abbr: REF5
  ref: ad2
```

Enhancements:

- Improved error messaging has been implemented including the start of QR-based error codes where more info is available on the website.
- Animation added when Weather engine is loading

Bug Fixes:

- Version numbers display properly on About page

5.0.0 (2025-03-30)

5.0.0 is a substantial under-the-hood rewrite that updates every component in the system to upgrade the underlying frameworks TVS is built upon.

BREAKING CHANGES:

- Removed incompatible Vime library. `video@1` and `video@2` have been removed and should not be used anymore. To fix your config file if it's broken, use `video` rather than `video@1` or `video@2`.
- `hlsBugFix` option was removed from `video` engine since the bug that it was fixing isn't present anymore.
- Video game `console` option is no longer recognized; use `core` instead (https://nostalgist.js.org/apis/launch/#core)
- The "1989" theme should be specified in quotes.
- Disabled Nightwave Plaza API querying due to CORS blocking.

Replace this:

```
guide:
  theme: 1989
```

To this:

```
guide:
  theme: "1989"
```

Dependency Updates:

- Updated to Svelte 5 and rewrote all content engines to include better type definitions.
- Updated Nostalgist library to 0.14 from 0.11 to fix some bugs in the VideoGame engine.

Bug Fixes:

- Fixed invalid HTML structure in program guide tables
- Added better teardown logic for exiting video games
- Addressed issue where channels were not properly torn down upon channel change
- Fixed audio volume issues with Visualizer engine
- Fixed aspect ratio issue in Bouncing Logo engine
- Fixed verbiage for off-air channels in Channel List engine

New Features:

- Added schema validation support for configuration files. To take advantage of this, add the following line to the top of your configuration file, including the "# " at the beginning:

```
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.0.0.json
```

Then open the file in Visual Studio Code with the YAML extension installed.

When new versions of the schema are released I'll call them out in the notes; you'd change the `5.0.0.json` to the latest schema version to validate against it instead.

- Added `stretch` aspect ratio to the Image engine. This will stretch images to the size of their container regardless of aspect ratio.
- Added `shuffle` parameter to loops that allows you to shuffle them randomly or in such a way that the entire cycle gets played through once, then repeated again.

```yaml
- number: 181
  name: Shuffled Loop
  abbr: LSHUF
  loop:
    shuffle: random # You can shuffle the content in a loop. Use `random` to shuffle the content randomly, or `cycle` to shuffle the content in a cycle.
    content:
    - image: placeholders/sequence/0.jpg
      duration: 3
    - image: placeholders/sequence/1.jpg
      duration: 3
    - image: placeholders/sequence/2.jpg
      duration: 3
    - image: placeholders/sequence/3.jpg
      duration: 3
    - image: placeholders/sequence/4.jpg
      duration: 3
    - image: placeholders/sequence/5.jpg
      duration: 3

- number: 182
  name: Cycled Loop
  abbr: LCYCL
  loop:
    shuffle: cycle # Cycle shuffling will play back the content in a random order but show all content before repeating.
    content:
    - image: placeholders/sequence/0.jpg
      duration: 3
    - image: placeholders/sequence/1.jpg
      duration: 3
    - image: placeholders/sequence/2.jpg
      duration: 3
    - image: placeholders/sequence/3.jpg
      duration: 3
    - image: placeholders/sequence/4.jpg
      duration: 3
    - image: placeholders/sequence/5.jpg
      duration: 3
```

Aesthetics:

- Updated the default welcome image with a new look.

4.5.0 (2025-01-27)

Bug fixes:

- Image aspect ratio behavior was incorrectly called `behavior` and now is called `aspectRatioBehavior`.
- Overflow scrollbars are now hidden if content overruns the screen
- We now close an audio context when destroying it, which should help prevent issues with multiple contexts building up. Raspberry Pi 5 is confirmed to play audio better now.

New features:

- New OSD tuner theme based on mid-2010's satellite TV: `sat2012`! Here's an example:

```
osd:
  theme: sat2012
  aspectRatioBehavior: stretch # Optional. The aspect ratio behavior of the screen. Values are stretch, letterbox, or hybrid. Defaults to hybrid.
  # primaryColor: red
  # accentColor: black
  themeOptions: # certain themes have additional options that can be set like this:
    providerLogo: placeholders/no-signal.png # Optional. The provider logo to show in the info OSD. If not set the default logo will be shown. If set to false no logo will be shown.
    extraMenuItemsBefore: # Non-functional extra menu items can be added for asthetic purposes.
      - Messages
    extraMenuItemsAfter: # More non-functional menu items.
      - CC
      - Last 4
      - Favorites
      - Audio/Video
      - Parental
```

- Added a new about page to show the program version, credits, dependencies and special thanks. Press `alt + v` to go there.

4.4.0 (2025-01-12)

Enhancements:

- Images can now be constrained to aspect ratios if they don't fill the screen.

```
- number: 2
  name: Welcome
  abbr: HELLO
  image:
    src: placeholders/welcome.png # If you use more than one option for images you need to specify the source like this.
    aspectRatioBehavior: contain # Optional. The aspect ratio behavior of the image. Values are contain or cover. Defaults to cover.
```

New features:

- Ability to play background audio in the "no signal" channel

```
noSignalContent: # Optional. You can specify content to show when the TV is off the air.
  image: placeholders/no-signal.png # Required. The image to show when the TV is off the air.
  backgroundAudio: placeholders/bgm-3.mp3 # Optional. You can add background audio to the no signal screen.
```

- Added ability to specify custom screen dimensions in the config file. After modifying this value it will apply when your picture settings are reset.

```
# Screen dimensions defaults can be overridden here. If you don't specify these the TV will be 640x480 by default.
# Not all channel content will scale to fit the screen; some content will always render at its native resolution (640x480)
# **Note**: If you change these values, be sure to reset your picture settings using `alt + r` or you won't see the changes.
dimensions:
  screenWidth: 1280 # Optional. The width of the screen in pixels.
  screenHeight: 720 # Optional. The height of the screen in pixels.
  # scaleX: 100 # Optional. The scale of the screen percentage-wise on the X axis. Default is 100.
  # scaleY: 100 # Optional. The scale of the screen percentage-wise on the Y axis. Default is 100.
```

- Added customizable OSD behavior when displaying on different screen sizes:
  - Hybrid (default): centers the OSD where a standard 4:3 screen would be, with tasteful overflow for background graphics.
  - Stretch: fills the entire available space with the OSD.
  - Letterbox: centers and crops the OSD to 4:3 screen.

```
osd:
  theme: sat2007
  aspectRatioBehavior: stretch # Optional. The aspect ratio behavior of the screen. Values are stretch, letterbox, or hybrid. Defaults to hybrid.
  themeOptions: # certain themes have additional options that can be set like this:
    showMessageIcon: true
    showChannelLockIcon: true
```

4.3.0 (2025-01-05)

Bug fixes:

- Fixed bug with TVS overrunning the screen by a pixel or two on certain screen sizes triggering scroll bars.
- Fixed bug with background audio that would cause audio to sometimes fail on page load.

- Fixed layout bug in examples page
- Fixed default program guide font on examples page
- Fixed bug involving screen bezel size on hard page reload

Enhancements:

- Gave OSD a very slight blur effect when global blurring is on.

New features:

- Added the ability for channels to have icons.

```
- number: 44
  name: 44cast
  abbr: WX
  icon: placeholders/weather.png # Optional. Icon to show in the info OSD (if
applicable theme applied)
```

- Custom “no signal” overrides for channels without content.

```
noSignalContent: # Optional. You can specify content to show when a channel
is off the air.
  image: placeholders/no-signal.png
```

- New OSD theme based on mid-2000’s satellite TV: `sat2007`! Here’s an example:

```
osd:
  theme: sat2007
  # primaryColor: "rebeccapurple" # Optional color overrides
  # accentColor: "white"
  themeOptions: # certain themes have additional options that can be set like
this:
    showMessageIcon: true # Show the message icon in the top bar
    showChannelLockIcon: true # Show the channel lock icon in the top bar
```

- New engine that shows a channel slideshow: `channel-list`. Here’s an example for it:

```
- number: 171
  name: Custom Channel List
  abbr: CLIST
  channel-list: # All options are optional so if you don't use any, use "true" to make it work.
    # channelRange:
    #   min: 30 # Optional. The channel to start the channel list with. No channels will be displayed with a lower number than this if specified.
    #   max: 49 # Optional. The channel to end the channel list with (inclusive). No channels will be displayed with a higher number than this if specified.
    primaryColor: cyan # Optional. The channel number text color.
    textColor: black # Optional. The text color for everything else.
    # showAs: abbr # Optional. The channel list can show abbreviations or names. Defaults to `name`
    # sort: number # Optional. The channel list can be sorted by number or name. Defaults to `number`
    # columns: 3 # Optional. The number of columns to display in the grid. Defaults to 2.
    # duration: 10 # Optional. The duration in seconds before advancing to the next page. Defaults to 10.
    backgroundImage: false # Optional. You can select a different background image. If set to false no image will be displayed (good for mixing). If not set the default image will be displayed.
    logo: /placeholders/tv.png # Optional. Logo to show at the top of the channel list
```

More examples are available in the example `config.tvs.yml`.

4.2.2 (2024-12-22)

- Fixed weather icons not loading when in a subdirectory.
- Fixed issues where a long Feed Reader title or article would push the entire page off-screen; long titles now scroll horizontally.
- Updated Nostalgist.js library to 0.11.0
- Fixed issue in Video Game engine where `megadrive` wasn't a supported core despite appearing in documentation. `genesis` and `megadrive` now point to the same core.

- Fixed bug where webhook wasn't fired on channel up or down events.
- Mitigated bug in YouTube Playlist engine when playing private or removed videos that would crash the player by reloading the playlist from the next video.

Experimental:

- `length` property added to YouTube Playlist engine to specify the number of videos in the playlist manually without needing a YouTube API key for shuffled playlists. I'm still working on trying to figure out why shuffle isn't right and may change the way this engine works in the future.

```
- number: 17
  name: Youtube Playlist Shuffled
  abbr: YTSFL
  youtube-playlist:
    src: PLU1WjbywtZxEGdTPm-qB0KzPGkRPw6npl
    shuffle: true # Optionally you can shuffle playlists this way. You need to specify the playlist ID above using src if you include more options.
    length: 13 # Optionally manually specify the length of the playlist so that no API key is needed to determine it.
```

4.2.1 (2024-12-21)

- Fixed bug in weather engine where radar station was being used for observations when observation stations should be used instead.

4.2.0 (2024-12-19)

- Fixed bug in bouncing logo engine where the logo wouldn't draw on load.
- Fixed bug with audio blips on channel changes when muted
- Fixed screen stretching function so that the entire TV container stretches, not the inside contents of the container.
- Added channel permalinks. By default now TVS will populate the document hash with the channel number. When you refresh TVS the previous channel will be loaded. You can change the way this works by using `persistCurrentChannel: localstorage` to save the current channel to local storage instead of URL on

change, or `persistCurrentChannel: false` to use the old behavior of not saving the channel on change. Permalinks will work regardless.

- Added toggle for animated channel change. In the config file set `animateChannelChange: false` to disable the “shake” effect when the screen changes.
- Added cache busting for the config file, since often this file is often cached more aggressively than we want. To disable this feature you’ll need to edit the environment file under `_app/env.js`: use `export const env={"PUBLIC_USE_CACHE_BUSTER":"false"}` instead of true.
- Added webhook support for external integrations. This is for receiving data *from* TVS, not sending to it. TVS will call these webhooks when channel, volume or mute events occur to let your service know it. This is useful if you want to show the channel or volume information on an external display.
  - All webhooks are sent as GET requests.
  - The bracketed variables will be replaced at the time that the webhook is fired. If TVS is showing channel 12 then `channel={channel}` will become `channel=12`.

```
# webhooks: # Optional. You can set up webhooks to send data to external
services when certain events happen.
#   channelChange: http://your-server/receive_update?channel={channel}&manual=
{isManualInput} # Optional. This is the URL that will be called when the
channel changes. The channel number and whether it was a manual input will be
sent as query parameters.
#   volumeChange: http://your-server/receive_update?volume={volume} # Optional.
This is the URL that will be called when the volume changes. The volume level
will be sent as a query parameter.
#   muteChange: http://your-server/receive_update?isMuting={isMuting} #
Optional. This is the URL that will be called when the muting or unmuting. The
mute state will be sent as a query parameter.
```

4.1.0 (2024-12-02)

- Added ability to disable the on-screen display entirely, or parts of it.

```
osd: false # You can disable the on-screen displays entirely by setting
this to false.
```

```
# You can also disable parts of the OSD individually. The following options
are available:
osd:
  tuner: false # You can disable the tuner OSD by setting this to false.
  info: false # You can disable the info OSD by setting this to false.
  receiver: false # You can disable the receiver OSD by setting this to false.
```

4.0.1 (2024-11-23)

- Fixed zoom bug on small window sizes introduced in previous update.
- Beginning to update TVS's versions to use semantic versioning so that they show up correctly in GitHub releases and make it easier on users to tell when breaking changes occur.

4.0.0 (2024-11-21)

- Breaking change: guide alerts are attached to specific channels now; if you have a channel alert on channel 1 for instance but don't have a channel 1 in the lineup it won't be shown.
- Rewrote and restructured large parts of the Guide engine to support multiple themes. Two are supported as of this release: the default one and "1989", a classic single-slot guide in the style of the EPG (https://prevueguide.com/wiki/EPG_Sr.)
- Added the ability to filter guides by channel ranges.
- Modified the external listings data structure (TVSL JSON) to allow for more precise listings; listings are now measured by width in minutes and are not rounded down to the nearest timeslot anymore. Listings can fluidly span between guide timeslots in the default theme, and are displayed in a list in the 1989 guide separated by time. There's an example program available here that converts XMLTV to TVSL JSON: https://github.com/zshall/TelevisionSimulatorGuideData

- Added multiple guide examples to the default config file demonstrating custom colors and external listings (if you have a TVSL JSON file to parse).
- Changed the way that guide scrolling works to be more accurate; timeslot headers stick to the top of the guide until a new header replaces them.
- Changed the way that auto-scaling works to use `zoom` CSS instead of `transform` to mitigate bugs when scrolling.

```
- number: 170
  name: Guide 1989 full example
  noiseBlendMode: multiply
  guide:
    theme: 1989
    themeOptions: # Some themes have extra settings that can only be applied in these themes.
      title: Cable TV Guide # Optional. A title to show at the top of the guide
      ticker: # Optional. A scrolling text bar at the bottom of the guide. Put a list of messages you want shown here.
      - "Hypercom cable - connecting you to the future"
      - "Pay-Per-View Movies - Ch 1 - Straight from Hollywood to your home, anytime!"
    channelRange:
      min: 1 # Optional. The channel to start the guide with. No channels will be displayed with a lower number than this if specified.
      max: 100 # Optional. The channel to end the guide with (inclusive). No channels will be displayed with a higher number than this if specified.
```

3.5.0 (2024-10-17)

- Added `feed-reader` engine. This component will load an RSS feed and display articles one at a time. Articles should have a title and content.
- Added `color` engine. You can display a plain background with either a solid color or a CSS gradient.
- Added a new `mix` layout. You can use it to superimpose one component on top of another full-screen using any CSS mix mode and also vary transparency of either component (from `0.0` to `1.0`)

Here’s a simple example of the feed reader:

```
- number: 33
  name: Local News
  abbr: NEWS
  feed-reader: /placeholders/news.rss
```

A full example is in the `config.tvs.yml` demo file. There are a few new examples in there, including layouts and colors.

To make the RSS feed load properly make sure the server is sending it with CORS headers. If it isn't, you may need a locally hosted proxy such as cors-everywhere.

3.4.0 (2024-09-25)

Hotfix to address bug surfaced in newest Chromium version (129) related to overflow. TV screen is now `overflow: hidden`.

3.3.0 (2024-09-21)

- Added custom key remapping. You can now reassign keys to any button or combination of buttons supported by the browser.

Here's an example of how this works:

```
inputMapping: # Optional. You can override the default keyboard commands
here.
  volumeUp: q # For any key you override, the default key will be unassigned.
  volumeDown: a # The combos supported should be in lower-case; the keys
supported can be found at https://github.com/madrobby/keymaster?tab=readme-ov-
file#supported-keys
  power: shift+p, enter # Using + will require multiple keys to be pressed at
the same time. Using , will assign multiple combos to the same function.
  # Important: the browser may have some reserved key combos that can't be
overridden. If you're having trouble with a key combo, try another one.
```

The commands available to override are:

- 0

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- channelDown
- channelUp
- help
- info
- mute
- power
- resetPicture
- scaleDownX
- scaleDownY
- scaleUpX
- scaleUpY
- setup
- toggleAutoScale
- toggleBezels
- toggleBezelType
- toggleBlur
- toggleChangeChannelNoise
- toggleClosedCaptions
- toggleNoise
- toggleOSD
- toggleRemotePairingCode
- toggleScanlines

- `toggleShadowMask`
- `toggleShadowMaskType`
- `version`
- `volumeDown`
- `volumeUp`

3.2.1 (2024-09-15)

- PLS playlist file support added in `backgroundAudio` engine for tracks accessible via the web. An example playlist is provided in the placeholders directory.
- PLS files will provide info about what's now playing to the audio info engine.
- If you're using IIS on Windows to host the files, a new MIME type was added to the `web.config` file included for this purpose.

Here's the playlist example:

```
- number: 53
  name: Playlist Example
  abbr: PLS
  image: placeholders/channel-53.png
  backgroundAudio: placeholders/playlist-example.pls # You can use a playlist file to play audio. This can be a local file or a remote one but the contents need to be accessible via the web. Currently only PLS files are supported.
  backgroundAudioOptions:
    shufflePlaylist: true # Optional. If you want to shuffle the playlist set this to true. Only has an effect if a playlist file is used.
```

3.2.0 (2024-09-09)

- Added custom color support for the Guide engine.
- `guide` can have the following new options:

```
      guide: # Program guides read the current channel lineup to
reconstruct listings based off channel numbers, title and description.
https://greenhillplaza.notion.site/Program-Guides-
46e80919a3754efa9769f74dd00946fe?pvs=4
        primaryColor: white # The primary color of the guide; this is used for
the default background color of guide cells
        secondaryColor: blue # The secondary color of the guide; this is used
for the text color of headers and channel numbers
        textColor: black # The text color of the guide; this is used for the
text color of guide cells
```

- Refactored the color scheme library to be accessible for more purposes.

3.1.0 (2024-09-07)

- Power-off and power-on effects have been added; to disable, use `warmup: false` in the config file.
- To change the length of the warmup effect, set `warmup` to a number in seconds.
- Fixed more teardown code.

It looks like my bug report for Vidstack is being worked on, the workaround code will be able to be removed when that's done which should improve stability.

3.0.0 (2024-08-31)

Breaking changes

- Vidstack is now the default video provider. To use the deprecated Vime video provider use `video@1` instead of `video`.

Bug fixes

- To that end, the Vidstack provider has been cleaned up and a bug involving loading incorrect URLs with YouTube and Vimeo was fixed.

Please report any bugs you encounter using Vidstack since Vime will be removed at some point.

2.7.1 (2024-08-29)

New Features

- Background Audio Interrupts are now available.

```
      - video:
          src: https://www.youtube.com/watch?v=OlfL5TWWBtg
          interruptsBackgroundAudio: true # While any piece of content
designated as interrupting background audio is on-screen, the background audio
will be paused.
```

Opt-In

- Experimental new video engine is available using Vidstack instead of Vime. To use it change `video` components to `video@2`, like so:

```
  - video@2:
      provider: youtube
      src: OlfL5TWWBtg
```

- **Vidstack will replace Vime** as the default engine in the future, but I haven't tested it enough yet.
- Sunsetting Vime will occur in 3 stages:
  - Opt-in (currently as of 8-29-2024)
  - Opt-out (switch back to `video`, `video@1` and `video@2` will be deprecated)
  - `video@1` and `video@2` aliases will be removed, Vime will be removed and Vidstack will power `video` engine.
- Dailymotion support is unavailable in Vidstack.

Bug fixes and enhancements:

- Fixed some autoScale problems from yesterday regarding state changes when `autoScale` is set to false in the config file and hasn't been manually set yet.
- Added autoScale and a description of it to the `/setup` page.
- Added more agressive teardown code to combat a bug that continues to evade me regarding nested content: https://github.com/sveltejs/svelte/issues/5268
- Press `v` to see version information.
- Custom mouse cursor

2.7.0 (2024-08-28)

A few quality of life improvements today:

- Added full-screen toggle button to the touch menu. Long-press or click and hold on the TV to toggle this menu!
- Added status messages on all picture adjustments; when pressing a picture adjustment shortcut you'll get a status update on what you changed.
- AutoScale has a manual override toggle: Alt + 7

2.6.1 (2024-08-24)

Today's update is all about improved stability.

- Config examples have been updated to allow running in a subdirectory. It's finally possible!
- If you have an existing config file and want to convert it to run in a subdirectory, remove the leading slash from local content. `/placeholders/...` should just be `placeholders/...` for instance.
- Example files have a new parameter called `baseDirectory` which is only really useful for loading them from the `/examples` page. You likely won't need this parameter in your main `config.tvs.yml` file.
- YouTube videos were not respecting volume controls and now mute themselves now if the volume is set to 0.
- `autoScale` wasn't working properly as when set to false it would still resize due to how config files were loaded. Those problems are fixed now.

- About page now shows the TVS logo.

2.6.0 (2024-08-20)

- Added a Docker image for easier deployment.
- Added info button components - press `I` to open the info screen.
- Currently supported audio info providers are Icecast, Azuracast and a custom one for Nightwave Plaza. Let me know in the chat room if there's any other providers you'd like to see added!
- To make use of audio info you'll need to at minimum add the following background audio options:

```
  backgroundAudio: http://radio.plaza.one/mp3
  backgroundAudioOptions:
    description: Nightwave Plaza Radio # Optional. Credits will appear in the Info OSD
    nowPlayingUrl: https://api.plaza.one/status # Optional. If the radio service you're using has a song history API you can link to it here.
    nowPlayingProvider: plaza-one # Required if using nowPlayingUrl. Only a few providers are supported right now; see the documentation for more info.
```

```
  backgroundAudio: http://icecast.example.com/radio.mp3
  backgroundAudioOptions:
    description: Icecast example
    nowPlayingUrl: http://icecast.example.com/status-json.xsl
    nowPlayingProvider: icecast
```

```
  backgroundAudio: https://azuracast.example.com/listen/station_name/radio.mp3
  backgroundAudioOptions:
    description: Station Name
    nowPlayingUrl: https://azuracast.example.com/api/nowplaying_static/station_name.json
    nowPlayingProvider: azuracast
```

2.5.0 (2024-08-11)

- Fixed audio bug where channel changes that use same background audio source wouldn't be reloaded.
- Updated audio system with new granular controls: in Generator, Keno, Video, Visualizer, and YouTube Playlist engines you can set a `volume` parameter to a number between 0 and 1, or set it to false. Here's an example:

```
  video:
    volume: false # You can mute the video entirely if you want.
    provider: ...
    src: ...
```

- `BackgroundAudioOptions` section was added with new optional features for filtering, adjusting volume and adding white noise to the audio:

```
  backgroundAudio: ...
  backgroundAudioOptions:
    volume: 0.9 # You can adjust the volume of the background audio per-channel
    # corsMitigation: true # If you're hosting the app on a server that doesn't support CORS you can enable this to disable audio processing. The filter effects won't work but the audio should play (if CORS is the issue)
    filterType: highpass # You can apply a filter to the audio; values are lowpass, highpass, bandpass, lowshelf, highshelf, peaking, notch, allpass
    filterFrequency: 1000 # The frequency of the filter in Hz
    # noise: 0.08 # You can add white noise to the audio too
```

- Made a race condition bug in the video game engine less likely by destroying the component when it finishes loading if channel is changed while it loads.

- Added new Bouncing Logo engine which loads an image that bounces around the screen, changing its color whenever it hits the screen. The image should be transparent and a single color for the effect to look right.

```
- number: -2
  name: DVD Player
  bouncing-logo: https://upload.wikimedia.org/wikipedia/commons/e/e7/DVD-Video_Logo.svg # The bouncing-logo engine allows you to display an image that bounces around the screen in the style of a DVD logo screensaver.
```

2.4.0 (2024-07-27)

- Added another placeholder background music track
- Basic text is refreshed from data source when the last page is reached before going back to the first page
- Fixed bug where local audio wouldn’t loop
- Fixed but with HLS live streaming in Safari
- Fixed signaling issues with Slideshow engine
- Custom OSD color extends to remote pairing screen
- Support added for local and custom PeerJS servers
- Added touch gesture support using HammerJS
- Added some new configuration examples
- Added kiosk mode to disable certain features and an example configuration file

---

2.3.0 (2024-07-08)

- Added screen bevels; 2 different types are present by default. They can be enabled or disabled using “alt + 6” and the type can be changed with “shift + alt + 6”. You can also default this type via the config file:

```
bezel: flat
```

or

```
bezel: cylindrical
```

- Fixed the help page to add some missed keyboard shortcuts.
- Started the process of reorganizing resources so that eventually TVS will be able to run from a subdirectory.

2.2.0 (2024-07-07)

- Added Vimeo and Dailymotion provider support
- You no longer need to specify providers in many cases for videos; simply linking to a YouTube video or Dailymotion or Vimeo will detect the video ID.

2.1.0 (2024-07-02)

- Press F1 to view keyboard shortcuts; the close button in the window title bar will take you back to the TV or you can press back on your browser.
- Number keys on the numpad can be used to change channels as well as the number keys at the top of your keyboard now. The Keymaster library providing shortcuts was updated to a fork with extended functionality.
- Added Video Game engine that supports NES, SNES, Genesis and Game Boy emulation using Nostalgist.js; there's an example at the bottom of the provided config file.
- Line inputs (defined by negative channel numbers) are now hidden from the channel guide.

2.0.0-beta (2024-06-20)

- Fixed off-air channel display in guide
- Added digital cable box theme for receiver
- Removed fade transitions from default OSD theme
- Added new background music to placeholders directory and beta welcome image

---

2.0.0-alpha6 (2024-05-16)

- Added digital cable box theme for tuner. Coming soon is a digital receiver theme.

To use the digital tuner here's an example:

```
osd:
  tuner:
    theme: digital
```

2.0.0-alpha5 (2024-05-13)

- Ability to override specific OSD components. Example:

```
osd:
  # default settings
  primaryColor: "#00FFFC"
  accentColor: "#F600FF"
  tuner: # tuner-specific overrides
    primaryColor: red
    accentColor: black
  receiver: # receiver-specific overrides
    primaryColor: "#36FF00"
    accentColor: black
```

2.0.0-alpha4 (2024-05-12)

- Added ability to change On-Screen Display colors in the configuration file. Here's an example:

```
osd:
  primaryColor: "#00FFFC"
  accentColor: "#F600FF"
```

2.0.0-alpha3 (2024-05-11)

- Made the white noise on channel change less harsh.
- Added split layout example configurations to match the documentation.
- Added basic text advertisement example as well as default setting support (data structure not yet finalized)

2.0.0-alpha2 (2024-05-10)

Added white noise generator to play when TV has no signal, also added optional white noise to play whenever changing channels. Use keyboard shortcut `alt+5` to enable or

disable this effect, or set `changeChannelNoise: false` in the top level of your configuration file.

Fixed bug where background audio volume wasn't updating when we adjust volume.

2.0.0-alpha1 (2024-05-09)

Project renamed from TV Simulator 99 to Television Simulator '99.

H-Split and V-Split layouts added. Each one has two properties: A and B. Guide layout has been updated to make use of this by default. Should allow greater flexibility in designing channels, but you will need to update your guide to remove the `upperContent` section! This is ignored in this new build. There's an example under `config-examples/03-guide.tvs.yml` that shows it in detail.

Remote has been updated to look better and includes numeric buttons. A known issue I'm looking into is the rate limiting I've seen from PeerJS and the disconnection events. This may only be solved if I set up a new negotiation server but we'll see.

Added a new `applicationTitle` parameter that allows you to change the title of the page in configuration after it loads.

Contact me on Slack or through other channels if you want to talk about anything related to TVS. Thank you again for helping me test!

1.0.0 (2024-03-31)

v0.6.2 has been renamed to v1.0.0 as this is the last version of the program-guide-only simulator.

The final version of the classic program guide. Version 2+ are based on a complete rewrite of the program in a new framework.

0.6.2 (2023-02-06)

Applied bug fix for height not calculating properly.

0.6.1 (2018-09-30)

Replaced unavailable YouTube video with a new one.

0.6.0 (2017-05-03)

Guide is decoupled from the main TV component.

0.5.3 (2017-05-01)

Implemented R and ADULT ratings

0.4.0 (2017-05-01)

Support more messages from the `<ads>` section of the XML.

0.3.3 (2017-04-29)

Fixed marquee height bug in Safari.

0.3.2 (2017-04-29)

Hid YouTube IFrame player controls.

0.3.1 (2017-04-29)

Added mute button.

0.0.1-preview (2017-04-28)

Initial version of `TV Simulator 99` committed.

Get Started

Television Simulator *(TVS)* is a configurable web-based media frontend that simulates an analog TV, tuner, receiver and cable headend. It is a web-based application and needs to be hosted on a web server to be used. Here are three ways to get started:

- Via your favorite web server
- Through the Docker image
- New Using the Television Simulator Launcher

Compatibility

TVS is designed to run on all platforms that support the latest version of the Chromium browser. Browsers like Google Chrome, Brave, Edge, etc. will work. Firefox and Safari may run but support is not guaranteed.

Using a Web Server

You can run Television Simulator on any web server that serves static files. This includes Apache, Nginx, IIS, and more. TVS is distributed as a collection of web files (HTML, CSS, JS, etc.) and can be hosted anywhere you'd like. Some features like Twitch integration require TVS to run in a "secure context", meaning either via HTTPS or locally on your computer using `localhost`.

Windows

>
>
> **Heads up!**
>
> Try the new Television Simulator Launcher for an easier way to install, update, and manage TVS on Windows!

If you want to use a traditional web server on Windows instead, here's how to get started.

Download and unzip the web files

1. Get the latest **release** from GitHub.
2. Click the latest **`tvs-{version}.zip`** file to download it.
3. In **File Explorer**, right-click the zip → **Extract All…**
    - Pick a simple folder, e.g. `C:\TVS99`
    - When you open `C:\TVS99`, you should see files like `index.html`, `placeholders/`, etc. (If there's an extra top-level folder, go into it; the folder that contains `index.html` is what we'll serve.)

Use the built-in Windows web server (IIS)

Turn on IIS (one-time)

1. Press **Start**, type **Windows Features**, open **Turn Windows features on or off**.
2. Check **Internet Information Services** (leave default sub-items as they are) → **OK**.
3. Wait for it to install.

Point IIS at your folder

1. Press **Start**, type **IIS** and open **Internet Information Services (IIS) Manager**.
2. In the left tree, right-click **Sites** → **Add Website…**
3. Fill in:
    - **Site name:** TVS99
    - **Physical path:** `C:\TVS99` (the folder with `index.html`)
    - **Port:** `8080` (to avoid conflicts—any free port is fine)
4. Click **OK**. If it didn't auto-start, click **Start** on the right.
5. Open your browser and go to **http://localhost:8080/** You should see Television Simulator '99.

> To stop later: in IIS Manager, select **TVS99** → **Stop**. To remove it: right-click the site → **Remove**.

Troubleshooting

- **I only see a file list, not the app.** Make sure you're serving the folder that directly contains `index.html` (not a parent folder).
- **Firewall prompt.** If Windows asks, click **Allow** so your browser can connect to the local server.
- **Invalid configuration message displayed.** Make sure that YAML files are allowed to be served by IIS. By default, IIS blocks unknown file types. You can add a MIME type for `.yml` files with the value `text/yaml` in IIS Manager under your site → **MIME Types**.

Using Docker

Get the official TVS Docker image and run it in a container.

Linux

This is the recommended way to run TVS as a dedicated server. You get easy updates and can run multiple instances on different ports.

Install Docker Engine

I won't cover this in detail because it varies by distro. Follow the official instructions.

Docker Compose (recommended)

The Docker image exposes port 3000 as a static web server with all of Television Simulator installed. You need to provide your own configuration file however, which can be done with a volume. If you'd like to serve custom content from TVS server, create a content volume and use `/content/your-media.jpg`, etc. as the path in `config.tvs.yml`.

Here's an example `docker-compose.yml` you can use:

```
services:
  tvs-server:
    image: 'zshall/television-simulator:latest'
    restart: unless-stopped
    ports:
      - '80:3000'
    volumes:
      - /var/tvs/config.tvs.yml:/home/static/config.tvs.yml
      - /var/tvs/content:/home/static/content
```

Windows + Docker Desktop

Install Docker Desktop and set it to auto-start when you sign in

I'd recommend trying Linux for a dedicated server, but if you want to use Docker on Windows here's how to get started.

1. Install **Docker Desktop for Windows** (from docker.com).
2. Open Docker Desktop → **Settings** (gear) → **General** → turn on **Start Docker Desktop when you sign in to your computer**.

> Note: Docker Desktop starts **after** you log in unfortunately. Containers marked `restart: unless-stopped` will auto-start once Docker is running.

Make a folder structure

Create a folder for your setup, for example:

```
C:\TVS99\
├─ docker-compose.yml
├─ config\
│   └─ config.tvs.yml        ← your TVS config file
└─ content\                  ← any media you want TVS to serve
    ├─ images\
    └─ videos\
```

- If this is the first time you're using TVS you may want to start with the example config file. This means not mapping the volume for `config.tvs.yml` shown below and just using the default config that comes with the image to familiarize yourself with the app. You can always add your own later.
- Inside your config, you can reference files in the `content` folder using paths like **`/content/your-media.jpg`**.

Create `docker-compose.yml`

In `C:\TVS99\docker-compose.yml`, paste:

```
services:
  tvs:
    image: zshall/television-simulator:latest
    container_name: tvs99
    restart: unless-stopped
    ports:
      - "8080:3000"              # visit http://localhost:8080
    volumes:
      - ./config/config.tvs.yml:/home/static/config.tvs.yml:ro
      - ./content:/home/static/content:ro
```

Why these paths? The image serves the app from a static web server on **port 3000**, and it expects your config file to be available from the site root. Mapping to `/home/static/...` makes the file available at `http://localhost:8080/config.tvs.yml`, and mapping the folder to `/home/static/content` makes your content reachable at `/content/...` as referenced in your config.

> If you already run something on port 8080, change `"8080:3000"` to another free port like `"8081:3000"`.

Start it up

1. Open **PowerShell**.
2. Run:

```
cd C:\TVS99
docker compose pull
docker compose up -d
```

3. Open your browser to **http://localhost:8080/**. You should see Television Simulator ‘99.

Everyday use

- **Start with Windows sign-in:** Docker Desktop will launch after you log in (because you enabled that toggle). Your `tvs99` container will auto-start thanks to `restart: unless-stopped`.
- **Stop the app:**

```
docker compose down
```

- **Update to the latest image later:**

```
docker compose pull
docker compose up -d
```

Configuring TVS

Now that you've got TVS running, it's time to make it your own! Everything in TVS can be set up in a YAML-based config file named `config.tvs.yml`, located in the root of your installation directory. This file tells TVS what channels to show, where to find your media, and how everything should look.

Conceptual overview

Television Simulator simulates the following components of a traditional television experience:

- **TV Screen**: we simulate a television screen with configurable aspect ratios and screen effects, such as CRT scanlines, picture noise, picture blur and shadow masks. Standard definition, high definition and custom resolutions can be set. The screen will automatically resize to fill available space in your browser window and works in full screen as well. All of these effects can be disabled or customized, and depending on the hardware you're running TVS on you may want to disable some effects for better performance.

- **Tuner and Receiver**: the look and feel of the on-screen displays can be skinned to look like a typical built-in TV tuner and a few different models of set top boxes. Colors can be customized and program information can be shown on-screen (if available). Volume controls and muting indicators get their own set of on-screen controls as well. You can mix and match tuner and receiver themes.

- **Headend**: In the cable TV business, a headend is a facility where channels are received from different sources and combined into the channel lineup that's transmitted to you. Television Simulator lets you define your own channels by number and to decide what content to put on each channel. Most of your configuration file will likely consist of channels and content.

What's a YML file?

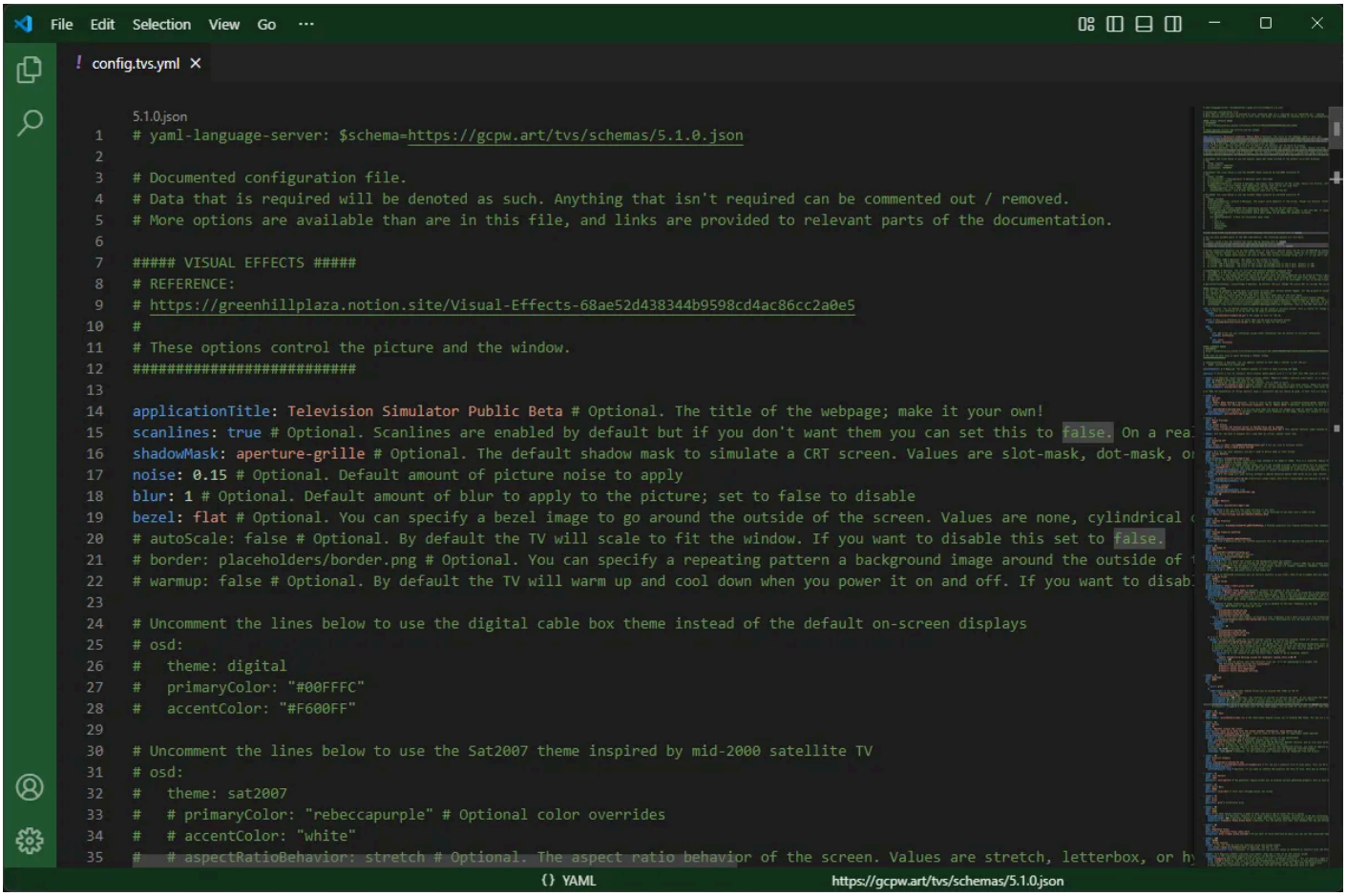
The TVS configuration file is written in the YAML language. It's not a programming language but rather a way to represent data in a human-readable format. YAML is often used for configuration files because it's easy to read and write. To make it even

easier to read and write, you can use a program like Visual Studio Code (known as VS Code) to edit your config file. VS Code can highlight your YAML syntax and help you avoid mistakes.

Configuring Visual Studio Code

To configure VS Code for editing YAML files, you can install the YAML extension from the Visual Studio Code Marketplace. This extension provides features like syntax highlighting, autocompletion, and validation for YAML files, making it easier to work with your TVS configuration.

> **Caution**
>
> The YAML extension has a known bug in version 1.18 that can lead to incorrectly saying that your config file is invalid. This should be fixed in version 1.19, but until then install version 1.17 of the extension.


File Edit Selection View Go ···
config.tvs.yml
5.1.0.json
yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json
Documented configuration file.
Data that is required will be denoted as such. Anything that isn't required can be commented out / removed.
More options are available than are in this file, and links are provided to relevant parts of the documentation.
VISUAL EFFECTS
REFERENCE:
https://greenhillplaza.notion.site/Visual-Effects-68ae52d438344b9598cd4ac86cc2a0e5
#
These options control the picture and the window.
applicationTitle: Television Simulator Public Beta # Optional. The title of the webpage; make it your own!
scanlines: true # Optional. Scanlines are enabled by default but if you don't want them you can set this to false. On a rea
shadowMask: aperture-grille # Optional. The default shadow mask to simulate a CRT screen. Values are slot-mask, dot-mask, o
noise: 0.15 # Optional. Default amount of picture noise to apply
blur: 1 # Optional. Default amount of blur to apply to the picture; set to false to disable
bezel: flat # Optional. You can specify a bezel image to go around the outside of the screen. Values are none, cylindrical
autoScale: false # Optional. By default the TV will scale to fit the window. If you want to disable this set to false.
border: placeholders/border.png # Optional. You can specify a repeating pattern a background image around the outside of
warmup: false # Optional. By default the TV will warm up and cool down when you power it on and off. If you want to disab
Uncomment the lines below to use the digital cable box theme instead of the default on-screen displays
osd:
theme: digital
primaryColor: "#00FFFC"
accentColor: "#F600FF"
Uncomment the lines below to use the Sat2007 theme inspired by mid-2000 satellite TV
osd:
theme: sat2007
primaryColor: "rebeccapurple" # Optional color overrides
accentColor: "white"
aspectRatioBehavior: stretch # Optional. The aspect ratio behavior of the screen. Values are stretch, letterbox, or h
{} YAML
https://gcpw.art/tvs/schemas/5.1.0.json

Automatic validation

In Visual Studio Code you can validate your config file with the YAML extension installed by making sure that it begins with the following text which points VS Code to the TVS *schema* file:

```
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json
```

As TVS evolves, the schema version may change. All previous schemas will remain available for backward compatibility.

Reading the example config

Television Simulator ships with an example configuration file (`config.tvs.yml`) that showcases different features. More example configurations are available in the `config-examples` directory, and can be previewed by navigating to `/examples` in TVS itself.

Tip

It’s a good idea to back up the example config file, as ultimately you’ll be editing or replacing it with your custom configuration.

Channels and content tutorial

Minimum example

At a minimum, your configuration file must have at least one channel defined. Channels must have unique channel numbers. To learn more, backup your `config.tvs.yml` file and replace it with the following contents:

```
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json

channels:

- number: 1
```

This configuration has a single channel, channel 1. It has no content so when you start TVS up with this config you'll see nothing but static. The `- number: 1` line begins defining the channel. Each channel will start with a line like this after the `channels:` block which starts the list.

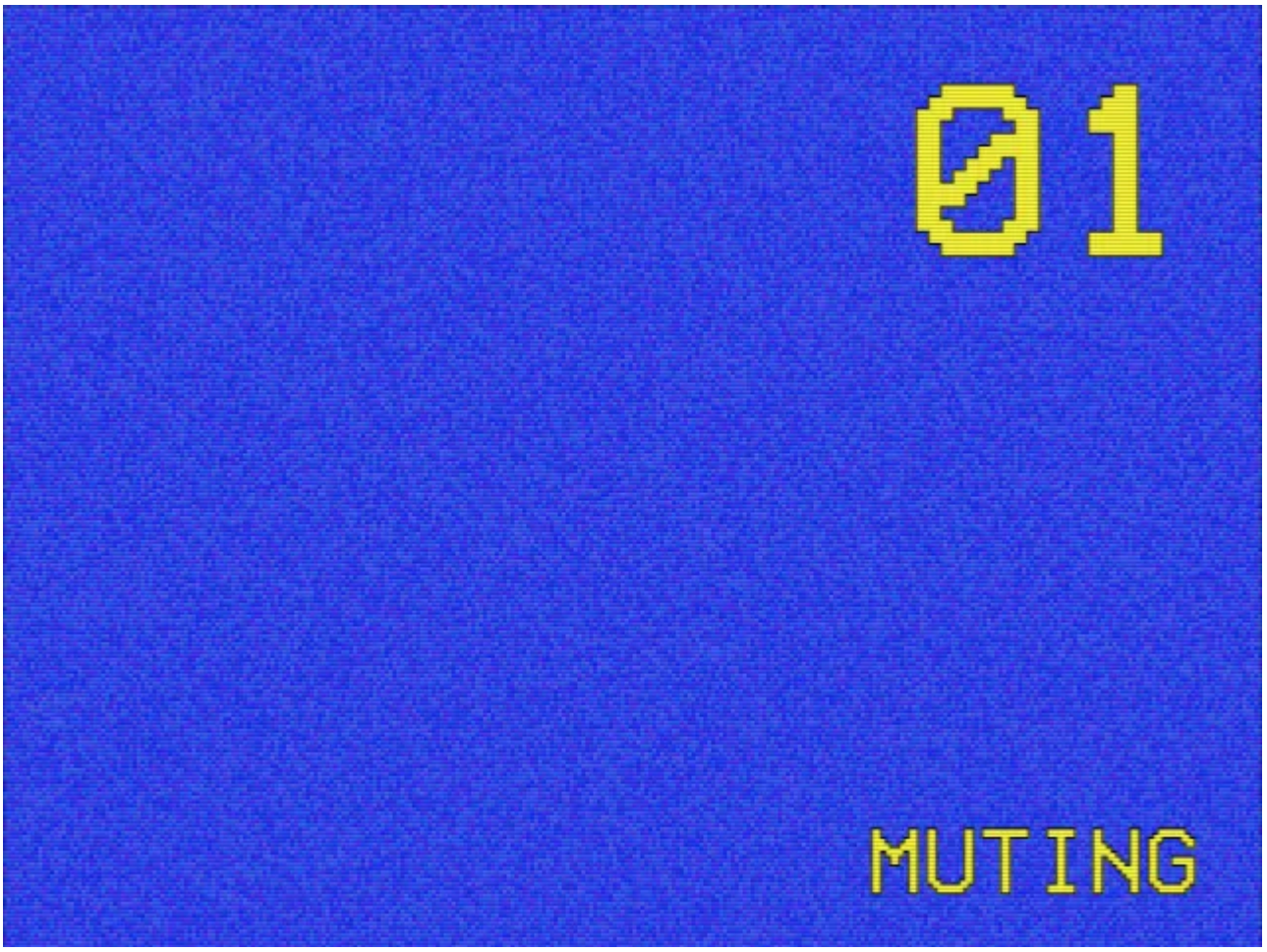
We can make this a bit more interesting by adding some content to the channel. **Channels can only have one top-level content item**, but this isn't as limiting as it sounds (we'll get into that later). Add some content like this:

```
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json

channels:

- number: 1
+  color: blue
```

Caution

Note the indentation! YAML requires consistent indentation to work correctly. Here we're using two spaces for indentation.

All lines that are indented under the `- number: 1` line are part of the channel's definition. Here we've added the `color` content engine, which simply displays a solid background color.

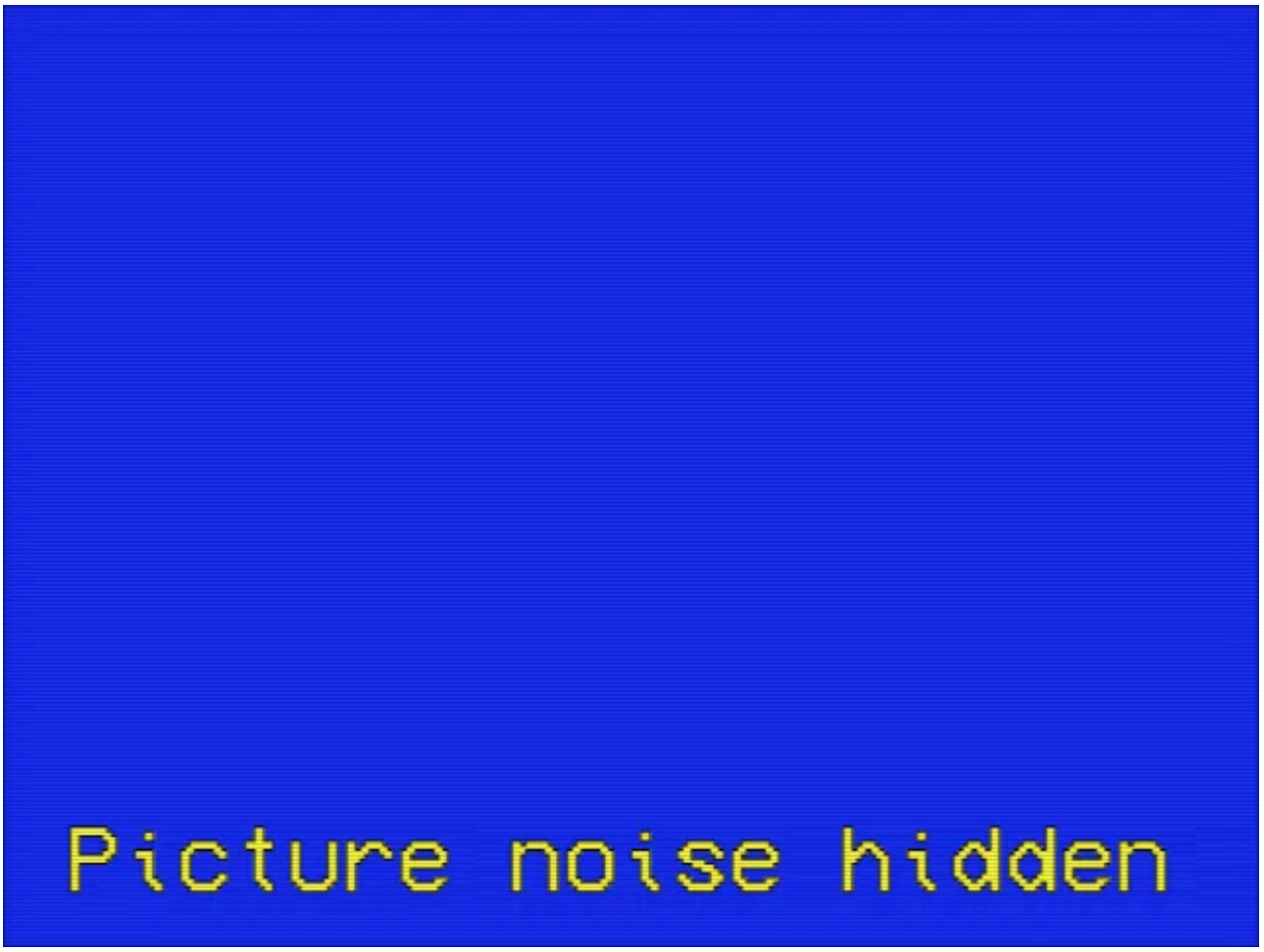


Defaults

As you can see in the screenshot, we're using a standard-definition picture with the default tuner and receiver themes which provide an old-school look. A bit of picture noise, scanlines and blur are also enabled by default. Say you'd prefer a more modern appearance, you can change the defaults in your configuration file (to change the defaults for everyone) or by using keyboard shortcuts (to change how it looks for your browser only). You may want to do this to find out how you'd prefer things to look, then set the defaults in your configuration file.

Picture controls

Let's try changing the way the simulated TV screen looks. With TVS running, press the keyboard shortcut `Alt2` `Alt2` to turn off picture noise. You'll see a message in the bottom of the screen showing that the picture changed:



Similarly, if you use `Alt1` `Alt1` you'll see the scanlines effect turn on and off. Alt commands from 1 through 7 are currently implemented and will change picture settings, so try them out to find out how they work!

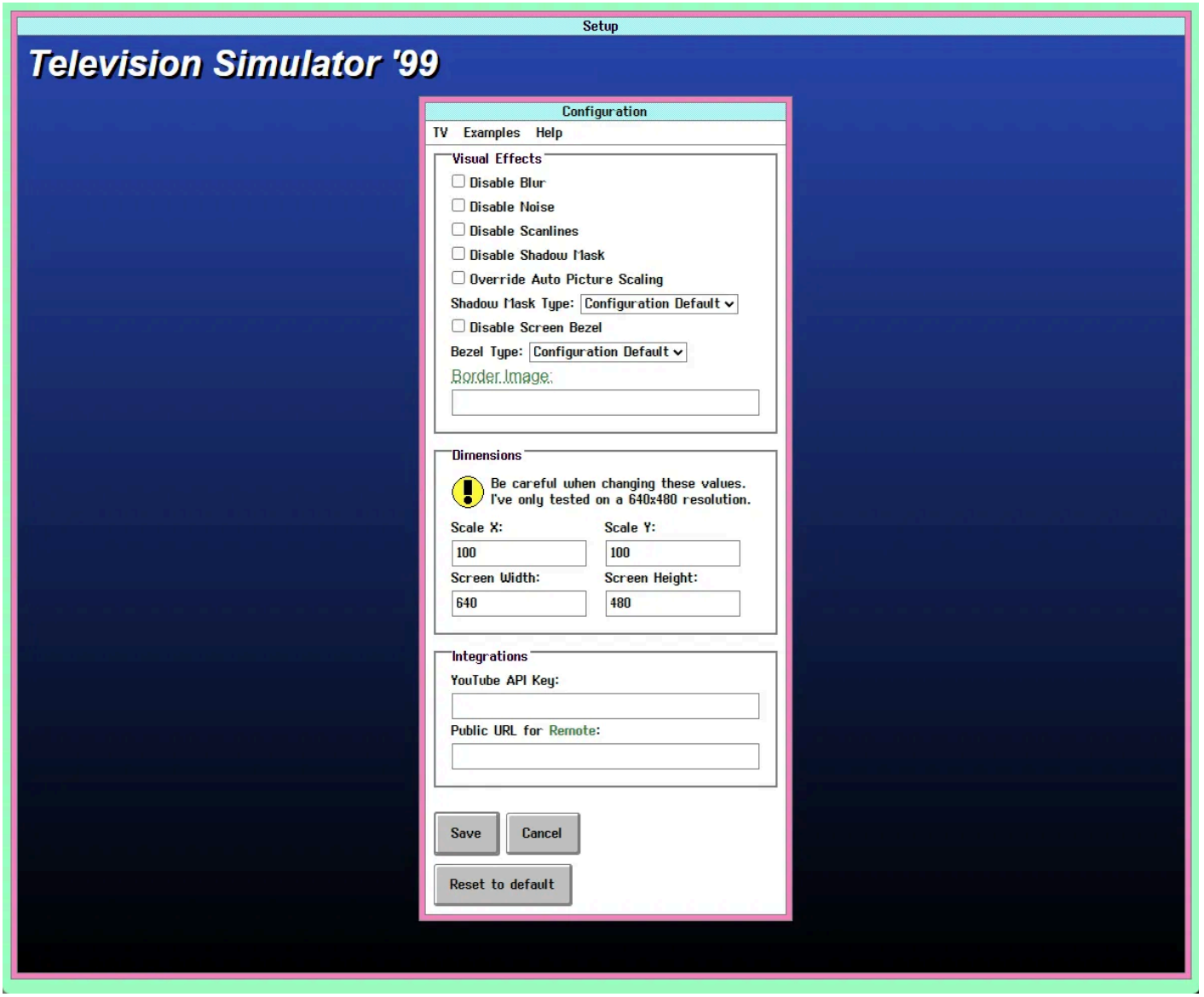
In the Keyboard Shortcuts section you can find a full list of keyboard shortcuts and what they do.

Resetting the picture

You can screw up the picture settings quite a bit, but don't worry! You can always reset to the defaults (as defined in the config file) by pressing `AltR` `AltR` . This will reset all picture settings and reload the page.

Graphical Setup

Instead of using keyboard shortcuts you can also use the graphical setup interface. Press `F4` `F4` to open the setup menu or go to `/setup` (example). These settings are only changed for your browser, not for any other users.


Setup
Television Simulator '99
Configuration
TV
Examples
Help
Visual Effects
Disable Blur
Disable Noise
Disable Scanlines
Disable Shadow Mask
Override Auto Picture Scaling
Shadow Mask Type:
Configuration Default
Disable Screen Bezel
Bezel Type:
Configuration Default
Border Image:
Dimensions
Be careful when changing these values.
I've only tested on a 640x480 resolution.
Scale X:
100
Scale Y:
100
Screen Width:
640
Screen Height:
480
Integrations
YouTube API Key:
Public URL for Remote:
Save
Cancel
Reset to default

Playing Background Audio

Background audio can be added to any channel by using the `backgroundAudio` property in the channel's configuration:

```
channels:

- number: 1
  color: blue # some example content
+ backgroundAudio: placeholders/bgm-3.mp3
```

In the example above we're playing a single MP3 file from the `placeholders` directory:

```
index.html
config.tvs.yml
placeholders
    bgm-3.mp3
```

This property is defined on the **channel**, not the channel's content, and will apply to the entire channel. You can only have one background audio track per channel.

Radio streams

To add more dynamic content, you can use internet radio streams as background audio. You can either link to a public radio stream or host your own using a program like Azuracast or Icecast.

Note

Please ensure you have the right to use any radio streams you include in your channel. Television Simulator can be used to play anything that is accessible via a public URL, but does not ensure compliance with copyright or licensing requirements. Especially if you're running a publicly available TVS instance you'll need to be cautious about the content you include.

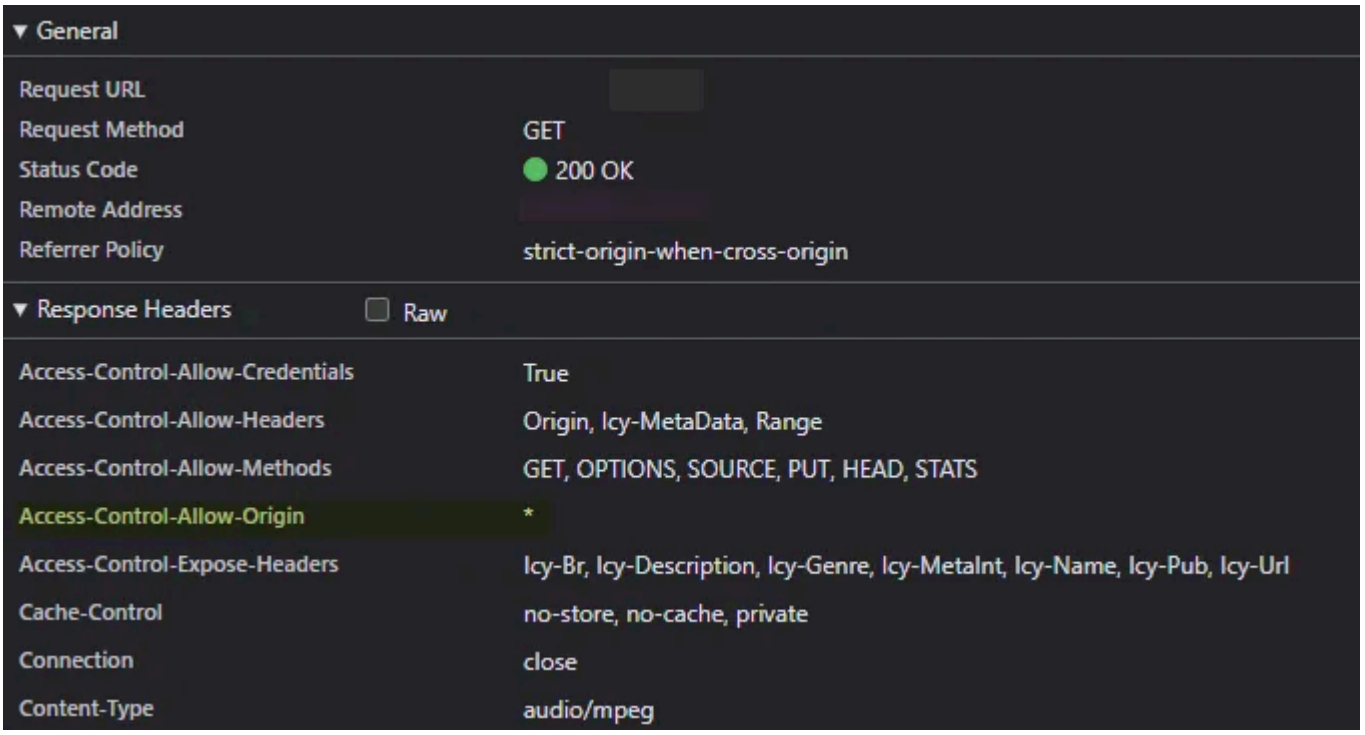
```
channels:

- number: 2
  channel-list: true # whatever channel content you want
+ backgroundAudio: http://radio.plaza.one/mp3
  backgroundAudioOptions:
    description: Plaza.one radio
```

Shoutout to Nightwave Plaza who has graciously allowed us to use their radio stream in TVS! Check them out and support them!

Breaking down what's happening here, we're linking directly to the stream itself, not an M3U8 or PLS file. The stream loads without any CORS errors because the server is configured to allow cross-origin requests.

The `backgroundAudioOptions` block has a `description` field where you can title the audio you're playing. This description will be shown in the info OSD (press `i i` on the keyboard to toggle this).

CORS error troubleshooting



`Access-Control-Allow-Origin: *` is a the key to avoiding CORS errors. This header tells the browser that it's okay to load this resource from a different origin (hostname).

Azuracast

Azuracast is what I'd recommend for making your own radio stations. Not only is it capable of hands-off 24/7 broadcasting, but it also provides an easy way to get information about what's playing which TVS can display. To do this, install Azuracast and set up a station. An easy way to get going is to turn on auto DJ using Liquidsoap, add music files and assign them to the `default` playlist and make sure that "Public Pages" are turned on. Suppose the mount point for your station is `/radio.mp3`, your configuration might look like this:

```
 channels:

 - number: 3
   channel-list: true # whatever channel content you want
+  backgroundAudio: http://my-azuracast-instance/listen/[station_name]/radio.mp3
   backgroundAudioOptions:
     description: My Azuracast stream
+    nowPlayingUrl: http://my-azuracast-
 instance/api/nowplaying_static/[station_name].json
+    nowPlayingProvider: azuracast
```

Icecast

Icecast is used by default in Azuracast but can be used on its own for a lighter-weight solution, though you'll need to roll your own broadcasting client. I won't get into the specifics of how to set a server up, but suppose you've got a single mount point (`/stream.mp3`) and an Icecast server running. Your config might look like this:

```
channels:

- number: 4
  channel-list: true # whatever channel content you want
+ backgroundAudio: http://my-icecast-instance/stream.mp3
  backgroundAudioOptions:
    description: My Icecast stream
+   nowPlayingUrl: http://my-icecast-instance/status-json.xsl
+   nowPlayingProvider: icecast
```

`status-json.xsl` is a file that lets you get information about what's playing in Icecast; this is what TVS reads to display the current track information.

>
> **Note**
>
> If you have multiple mount points and want to use metadata, you'll need to look inside `status-json.xsl` to find the `listenurl` and make sure you're using **that one** as your `backgroundAudio` URL, as that's how TVS knows which metadata to show.

Using a Playlist

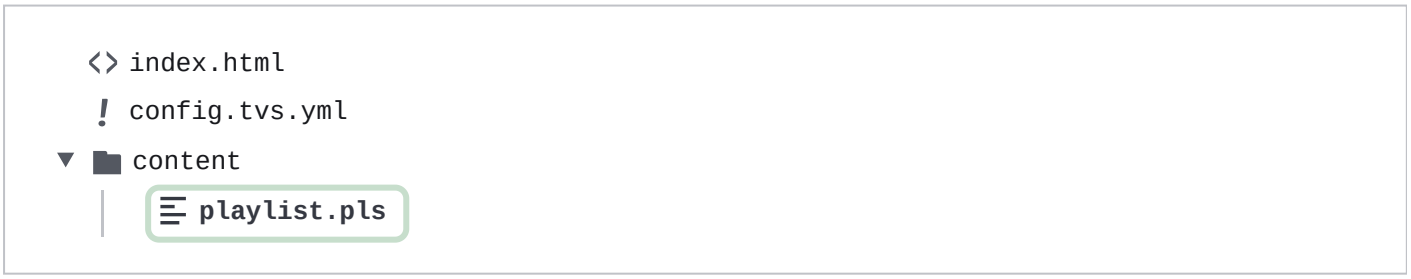
You can also use a PLS playlist as your background audio source. This is a more advanced setup as the playlist **needs** to resolve to audio files that are accessible over a web server.

Consider this example file:

```
[playlist]
NumberOfEntries=3
Version=2
File1=http://music-web-server/rock/artist1/song1.mp3
Title1=Song 1 - Artist 1
Length1=240
File2=http://music-web-server/rock/artist1/song2.mp3
Title2=Song 2 - Artist 1
Length2=300
File3=http://music-web-server/rock/artist2/song1.mp3
Title3=Song 1 - Artist 2
Length3=180
```

This will create a playlist with three songs, each with its own metadata (title and length). Make sure the audio files are accessible over the web and that the URLs are correct. You can then host this playlist file anywhere accessible to TVS over the network and use it as your background audio source:

```
channels:

- number: 5
  color: red # whatever channel content you want
+ backgroundAudio: content/playlist.pls
  backgroundAudioOptions:
    description: My PLS playlist
```



Using relative paths

Your playlist can use relative paths, but must be relative to the PLS file's location. Consider this setup on `http://music-web-server/`:

Your `playlist.pls` file might look like this:

```
[playlist]
NumberOfEntries=3
Version=2
File1=rock/artist1/song1.mp3
Title1=Song 1 - Artist 1
Length1=240
File2=rock/artist2/bonus.mp3
Title2=Bonus Track - Artist 2
Length2=300
File3=pop/artist3/single.flac
Title3=Single - Artist 3
Length3=180
```

Running as a Kiosk

On Linux (Raspberry Pi OS, etc.)

We'll be using Chromium-based browsers for this guide. To run Television Simulator in a full-screen kiosk, you can use the following command-line command:

```
chromium --kiosk --autoplay-policy=no-user-gesture-required --unsafely-treat-insecure-origin-as-secure --disable-web-security --user-data-dir=some-data-dir http://(your TVS URL)
```

>
>
> **Caution**
>
> For safety, don't run this as your main browser and be **careful** of the content you ingest. Don't use this to go just anywhere on the web.

Here's a breakdown of each option:

- `chromium`: the executable of the browser. Might be `google-chrome` or `brave`, `ungoogled-chromium` etc. (only you know which one you installed)
- `--kiosk`: Runs in kiosk mode, full screen with no menu bar or address bar. Use `commandq` `altf4` to quit.
- `--autoplay-policy=no-user-gesture-required`: Disables the autoplay restrictions that force TVS to start muted
- `--unsafely-treat-insecure-origin-as-secure`: Allows a regular HTTP connection to be treated as an HTTPS connection.
- `http:// your TVS URL`: the place TVS is hosted on your *local network*. Unless you trust a public instance and the people running it I would stick to a LAN IP (`http://192.168.1.123:8080` or whatever your IP is).

On Windows

The command does the same thing but it isn't quite the same syntax. Press `winr` `winr` to open the Run dialog, then paste the following command:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --kiosk --autoplay-policy=no-user-gesture-required --unsafely-treat-insecure-origin-as-secure --disable-web-security --user-data-dir="C:\tvs-kiosk-profile" http://localhost:3990
```

Before running, verify the following:

- Make sure Google Chrome is installed or change the path to the executable of your Chromium-based browser of choice.
- Change the URL at the end to your TVS URL. If you're running it on the same machine using the Windows Launcher, it defaults to `http://localhost:3990`
- The `--user-data-dir` path is a folder where the browser will save its data for the kiosk sessions. You can change it to whatever you want but make sure it's a folder that's not used for anything else.

Running on User Log In

If you want to run this command whenever you log in, you can add it to the Startup folder:

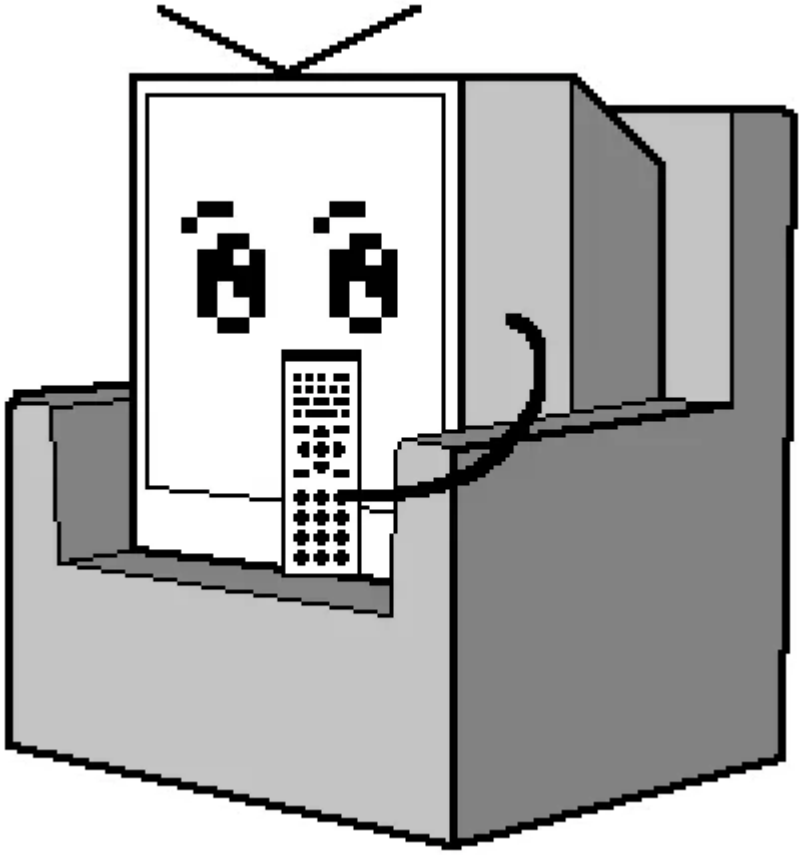
1. Press `winr` `winr` to open the Run dialog.
2. Type `shell:startup` and press Enter. This will open the Startup folder.
3. Right-click inside the folder, select New > Text Document, and rename it to something like `TVS Kiosk.bat`. Make sure to change the file extension from `.txt` to `.bat`.
4. Right-click the new `.bat` file, select Edit, and paste the command from above into the file. Save and close.

CORS Mitigation

If a web service you rely on has no CORS headers, you might run into issues when trying to access it from Television Simulator since it runs in a browser.

- `--disable-web-security`: Allows you to access resources which have CORS restrictions.
- `--user-data-dir=some-data-dir`: **Required if using `--disable-web-security`**: Chromium runs in an isolated sandbox to try and protect you; it'll save all user data somewhere else. Pick a folder path here and it looks like it'll be created if it doesn't exist.

Using the Remote



Television Simulator can be remote controlled using either the built-in Web Remote interface or a physical remote control.

Web Remote

The Web Remote is a feature in Television Simulator that allows you to control the TVS interface remotely using a web browser on another device, such as a smartphone or tablet. It's enabled by default and is peer-to-peer, only requiring an external server for initial connection setup.

The Web Remote uses WebRTC, and specifically the PeerJS library and its public signaling server.

Accessing the Web Remote

Pressing R R on your keyboard will display a QR code that you can use to access the Web Remote from another device. You can access the remote from your LAN IP address (for example, `http://192.168.1.x:3990/remote`).

If this doesn't work you can try using the public demo server at `https://tvs.gcpw.art/remote` to establish pairing. No data is stored or collected by the demo server; the connection is peer-to-peer between your devices.

>
> **Caution**
>
> If you're running TVS from `localhost` or `127.0.0.1` and using an older version of TVS (before 5.7.2) the QR code will be displayed but it won't work because these addresses are not accessible from other devices.

If you're running TVS 5.7.2 or later, you will see the pairing code displayed below the QR code.

If you didn't visit the QR code, you'll need to enter the pairing code manually. This will be a short code starting with `TVS-` followed by an identifier.

The identifier is unique to your TVS instance and tab; if you refresh the page it'll persist but if you close and reopen the browser tab a new code will be generated.

Using the Web Remote

Press the buttons on the Web Remote interface to control TVS as you would with a physical remote. If the connection is lost you can refresh the page and reconnect using the same pairing code.

Physical Remote

Television Simulator also supports physical remote controls provided the machine running TVS has a receiver of some sort (e.g., an IR receiver or Bluetooth) and can act as a standard keyboard.

One option that I recommend is the Flirc USB (not a sponsored link I just like it) which plugs into the front of your computer and can be programmed to work with almost any remote control as a keyboard.

You'll need to map the buttons on your remote to keyboard keys that TVS recognizes. You can find the list of supported keys in the Keyboard Shortcuts documentation.

Background Audio Options

Properties

`volume`

Type: `number` | `false`
Default: `1`

The volume level to use for playback, ranging from `0.0` (muted) to `1.0` (full volume). Setting to `false` or `0` will mute the audio.

`filterType`

Type: `'lowpass'` | `'highpass'` | `'bandpass'` | `'lowshelf'` | `'highshelf'` | `'notch'` | `'allpass'`

The type of audio filter to apply to the background audio. Leave blank to disable filtering.

`filterFrequency`

Type: `number`
Default: `0`

The frequency for the audio filter in Hz. Only applies when `filterType` is set.

`noise`

Type: `number` | `false`
Default: `0`

The amount of noise to add to the audio, ranging from `0.0` (no noise) to `1.0` (maximum noise). Setting to `false` disables noise.

`corsMitigation`

Type: `true` | `false`

Disable audio processing when the audio source doesn't have proper CORS headers. Note that filter options won't work when this is enabled.

`description`

Type: `string`

A description of the audio being played. Use the station name for live radio or song and artist information for music. This description will be shown in the info OSD (press `i` `i` to toggle).

`nowPlayingUrl`

Type: `string`

URL to fetch the now playing information from for dynamic content updates.

`nowPlayingProvider`

Type: `string`

The provider to use for fetching now playing information. Supported providers are:

- `azuracast`
- `icecast`

See the guide on background audio for more information.

`shufflePlaylist`

Type: `true` | `false`

Whether to shuffle the playlist if the background audio is a playlist. The default value is `false`.

Channels

Channels are the primary way to display content in Television Simulator. They can contain various types of content, including images, videos, and more. Each channel has a unique number and can be customized with metadata, visual effects and background audio.

Each channel should have at most one top-level content engine. If you don't define any content, the channel will be empty and TVS will display "no signal" content, which defaults to TV static. To display more content on a channel you can either use **layouts** to display multiple content items on screen at the same time, or **loops** to cycle between different content items, showing only one at a time.

Example

You need to define channels in a `channels` list in the YAML configuration file.

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE

- number: 124
  name: My second channel
  abbr: EXAMPLE2
```

To add content to a channel, figure out which content engine you want to use and add it as a property of the channel. For example, to add an image to channel 123:

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
+ image: /content/images/my-image.jpg
```

Program details

You can add program details to a channel, which tells the viewer what's currently playing. Currently only one program can be defined per channel. The program details include `title` and `description`. Program `title` will be displayed in the program guide and both `title` and `description` will be shown in compatible on-screen displays.

Background audio

Channels allow for background audio playback. You can specify a URL to an audio file that will play while the channel is active. This is useful for channels that display static images or other content that benefits from background music or sound. Background audio is separate from the channel's main content and can be controlled independently, but can be interrupted when particular content is being displayed.

Channel properties

number `required`

Type: `number`

The channel number. Must be unique across all channels. Negative channels are allowed and will be treated as line inputs. Channel `0` is displayed as "MENU". You can skip numbers; when pressing channel up or down TVS will skip numbers.

---

abbr

Type: `string`

A short abbreviation for the channel. Will be displayed in program guides and in OSDs that support it. If not provided it may show up as "UNKNOWN".

---

name

Type: `string`

The full name of the channel.

`title`

Type: `string`

The title of the currently playing program.

`description`

Type: `string`

A short description of the currently playing program.

`icon`

Type: `string`

The URL of an icon to display for the channel. In some OSDs this icon will display when the info screen is visible.

`blur`

Type: `number`

The amount of blur to apply to the channel’s video output, in pixels. Overrides the global blur setting.

`noise`

Type: `number`

The amount of noise to apply to the channel’s video output, from `0.0` to `1.0`. Overrides the global noise setting.

`noiseBlendMode`

Type: `normal` | `multiply` | `screen` | `overlay` | `darken` | `lighten` | `color-dodge` | `color-burn` | `hard-light` | `soft-light` | `difference` | `exclusion` | `hue` | `saturation` | `color` | `luminosity` | `plus-darker` | `plus-lighter`

The CSS blend mode to use for the noise effect. Defaults to `screen`.

---

backgroundAudio

Type: `string`

The URL of an audio file to play in the background while the channel is active.

---

backgroundAudioOptions

Options for customizing the background audio playback.

volume

Type: `number` | `false`
Default: `1`

Volume level from `0.0` (muted) to `1.0` (full volume). If set to `false` or `0`, the audio will be muted.

filterType

Type: `'lowpass'` | `'highpass'` | `'bandpass'` | `'lowshelf'` | `'highshelf'` | `'notch'` | `'allpass'`

Type of audio filter to apply; if none is specified, no filtering will be applied.

filterFrequency

Type: `number`
Default: `0`

The frequency, in Hz, for the audio filter.

noise

Type: `number` | `false`

Default: `0`

Amount of noise to add to the audio, from `0` (none) to `1` (maximum). Set to `false` to disable.

`corsMitigation`

Type: `true` | `false`

If enabled, disables audio processing for sources without proper CORS headers. Filter options will not work if this is set.

`description`

Type: `string`

A description of the audio source. Use the station name for live radio, or song and artist for music.

`nowPlayingUrl`

Type: `string`

URL to fetch “now playing” information.

`nowPlayingProvider`

Type: `string`

Provider to fetch “now playing” information from. See documentation for supported providers.

`shufflePlaylist`

Type: `true` | `false`

If the background audio is a playlist, enables shuffling when set to `true`.

Compatibility

TVS is designed to run on all platforms that support the latest version of the Chromium browser. Browsers like Google Chrome, Brave, Edge, etc. will work. Firefox and Safari may run but support is not guaranteed.

> **Tip**
>
> As of publication, Baseline 2023 web features are observed. If you're using an out of date browser you may encounter problems.

Browser Configuration

For the best user experience, you'll likely need to change permission settings to allow autoplaying audio and video. Since the project is designed around flipping through channels, clicking a play button would make it less enjoyable, therefore autoplay is essential.

Without modifying any browser settings, the simulator starts off muted and you must press M to unmute, which hopefully will meet the browser's threshold for interaction to allow videos and audio to play. Similarly, to connect a phone remote you must press R to display the QR code, which allows audio to play unmuted.

Keyboard Shortcuts

Press `F1 F1` to view the interactive keyboard help page.

Category	Combo	Description	Override ID
General	`P P`	Power	`power`
General	`R R`	Toggles display of Remote pairing code	`toggleRemotePairingCode`
General	`F1 F1`	Help (this page)	`help`
General	`F4 F4`	Go to the Setup page	`setup`
General	`V V`	Displays version information	`version`
General	`AltV AltV`	About TVS	`about`
Picture Adjustments	`AltR AltR`	Reset Picture	`resetPicture`
Picture Adjustments	`Alt1 Alt1`	Toggle Scanlines visual effect	`toggleScanlines`
Picture Adjustments	`Alt2 Alt2`	Toggle Noise visual effect	`toggleNoise`
Picture Adjustments	`Alt3 Alt3`	Toggle Blur visual effect	`toggleBlur`
Picture Adjustments	`Alt4 Alt4`	Toggle Shadow Mask visual effect	`toggleShadowMask`
Picture Adjustments	`Alt5 Alt5`	Toggle Static on channel change	`toggleChangeChannelNoise`
Picture Adjustments	`Alt6 Alt6`	Toggle Bezels visual effect	`toggleBezels`

Category	Combo	Description	Override ID
Picture Adjustments	`Alt7 Alt7`	Toggle automatic screen scaling	`toggleAutoScale`
Picture Adjustments	`ShiftAlt4 ShiftAlt4`	Toggle Shadow Mask Type	`toggleShadowMaskType`
Picture Adjustments	`ShiftAlt6 ShiftAlt6`	Toggle Bezel Type	`toggleBezelType`
Picture Adjustments	`ShiftAltLeft ShiftAltLeft`	Scales screen horizontal, negative	`scaleDownX`
Picture Adjustments	`ShiftAltRight ShiftAltRight`	Scales screen horizontal, positive	`scaleUpX`
Picture Adjustments	`ShiftAltUp ShiftAltUp`	Scales screen vertical, positive	`scaleUpY`
Picture Adjustments	`ShiftAltDown ShiftAltDown`	Scales screen vertical, negative	`scaleDownY`
Tuner	`Up Up`	Channel Up	`channelUp`
Tuner	`Down Down`	Channel Down	`channelDown`
Tuner	`I I`	Info	`info`
Tuner	`0 0 Numpad0 Numpad0`	Number 0	`numpad0`
Tuner	`1 1 Numpad1 Numpad1`	Number 1	`numpad1`
Tuner	`2 2 Numpad2 Numpad2`	Number 2	`numpad2`
Tuner	`3 3 Numpad3 Numpad3`	Number 3	`numpad3`
Tuner	`4 4 Numpad4 Numpad4`	Number 4	`numpad4`
Tuner	`5 5 Numpad5 Numpad5`	Number 5	`numpad5`
Tuner	`6 6 Numpad6 Numpad6`	Number 6	`numpad6`

Category	Combo	Description	Override ID
Tuner	`7 7 Numpad7 Numpad7`	Number 7	`numpad7`
Tuner	`8 8 Numpad8 Numpad8`	Number 8	`numpad8`
Tuner	`9 9 Numpad9 Numpad9`	Number 9	`numpad9`
Receiver	`Left Left`	Volume Down	`volumeDown`
Receiver	`Right Right`	Volume Up	`volumeUp`
Receiver	`M M`	Mute	`mute`
Video Game Controls	`E E`	Reset Console	-
Video Game Controls	`U U`	Up	-
Video Game Controls	`H H`	Left	-
Video Game Controls	`J J`	Down	-
Video Game Controls	`K K`	Right	-
Video Game Controls	`D D`	Start	-
Video Game Controls	`F F`	Select	-
Video Game Controls	`X X`	A	-
Video Game Controls	`Z Z`	B	-

Category	Combo	Description	Override ID
Video Game Controls	`S S`	Y	-
Video Game Controls	`A A`	X	-
Video Game Controls	`Q Q`	L	-
Video Game Controls	`W W`	R	-

Overriding Keyboard Shortcuts

In your configuration file you can override any keyboard command using the `inputMapping` section. For any key you override, the default key will be unassigned.

```
inputMapping:
  volumeUp: q
  volumeDown: a
  channelUp: alt+up
  channelDown: alt+down
```

Note

The browser may have some reserved key combos that can’t be overridden. If you’re having trouble with a key combo, try another one.

The combos supported should be in lower-case; we use Keymaster’s syntax and key names. Use the “Override ID” from the table above to find the command you want to override.

Remote Configuration

Using the `remote` section in your config file you can enable / disable the Web Remote or change the PeerJS server it uses for negotiation.

```
remote:
  enabled: false
```

```
remote:
  enabled: true
  server: 'your.peerjs.server.com'
  port: 9000
```

Properties

`enabled`

Type: `boolean`

Description: Whether to enable the Web Remote feature. The default value is `true`.

`server`

Type: `string`

Description: The PeerJS server URL to use for negotiation. The default value is the PeerJS cloud server.

`port`

Type: `number`

Description: The PeerJS server port to use for negotiation. The default value is `9000`.

`path`

Type: `string`

Description: The PeerJS server path to use for negotiation. The default value is `'/'`.

`peerId`

Type: `string`

Description: The PeerJS ID to use for this TVS instance. The default value is a random ID in the format `TVS-{random value}`.

>
>
> **Danger**
>
> Avoid setting a static `peerId` unless you have a specific reason to do so (like running a single instance on a known TV), as this can lead to connection issues if multiple TVS instances use the same ID.

Content Engine Primer

Tip

Read this first to understand how to use the reference documentation effectively!

Examples

Each content engine has its own page, and they are structured similarly. We begin with a brief summary and a screenshot followed by YAML code that shows how to use the engine in a channel.

For instance, you might see something like this:

```
image: /content/images/my-image.jpg
```

To use it, put it in a channel:

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  image: /content/image/my-image.jpg # <-- the content engine goes here
```

Or in a layout:

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  loop:
  - image: /content/images/my-image.jpg # <-- here
    duration: 10
  - image: /content/images/my-image-2.jpg
    duration: 5
```

Or even in a reference:

```
refs:
  ad1:
    image: /content/images/my-image.jpg # <-- here

channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  ref: ad1
```

Properties

After the examples you'll get an exhaustive list of *unique* properties for each content engine. These properties can be used to customize the behavior and appearance of the engine in your channels and layouts.

Properties that can apply to any engine won't be displayed on every page but instead are available below:

`_component` section

Type: `object`

Since: TVS 5.11.0

Beginning in TVS 5.11.0, the `_component` property groups options that can apply to any content engine. Right now it contains the `dimensions` property, which lets you define a virtual video signal for the component.

This is useful when you need to render content for a specific target resolution while still letting the containing channel, layout, or parent component scale it to fit the available space.

```yaml
generator:
  type: test-pattern
  _component:
    dimensions:
      width: 1280
      height: 720
      aspectRatioBehavior: contain
```

`_component.dimensions`

Type: `object`

Defines the size of the component's internal work area in pixels. Think of this as a simulated video signal that the component renders into before it is scaled into its container.

`width`

Type: `number`

Width of the work area in pixels.

`height`

Type: `number`

Height of the work area in pixels.

`aspectRatioBehavior`

Type: `'stretch' | 'cover' | 'contain'`

Default: `contain`

Controls how the virtual video signal is fitted into its container, which may be the root channel area, a layout region, or another component.

- `stretch`: Warps the signal to the exact size of the container. This can distort the aspect ratio.

- `cover`: Scales the signal until the entire container is covered. This can crop the image.
- `contain`: Scales the signal to fit entirely within the container. This can add letterboxing or pillarboxing.

`interruptsBackgroundAudio`

Type:: `true` | `false`

Whether the component takes precedence over the channel's background audio and mutes it while being displayed.

Shorthand Properties

`shorthand` properties are marked in blue and are properties that can be set using shorthand syntax; instead of saying:

```
image:
  src: /content/images/my-image.jpg
```

you can simply write:

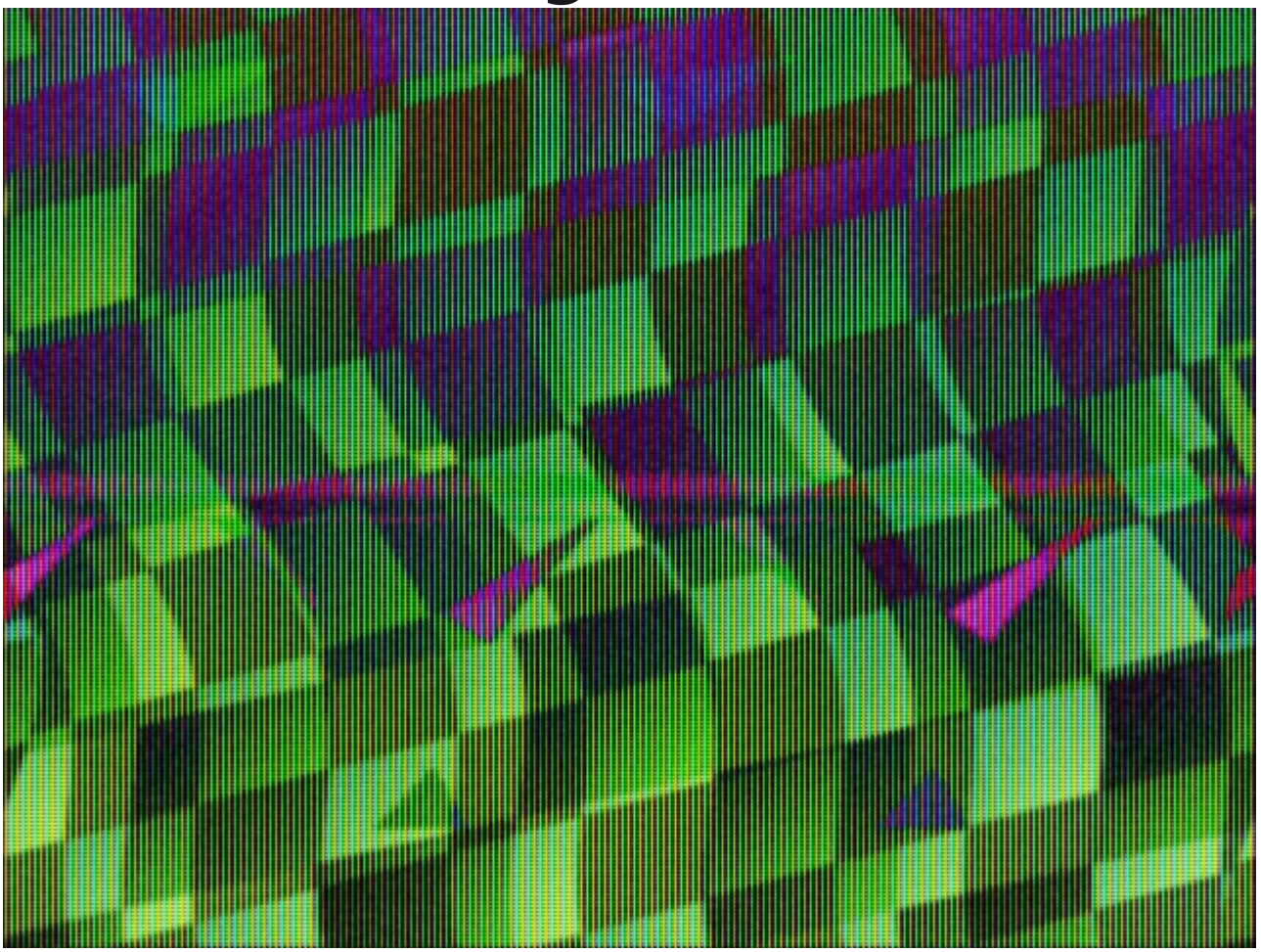
```
image: /content/images/my-image.jpg
```

You can't provide any other options with this shorthand syntax, but it is useful for simple cases such as when you just want to display an image or a video without any additional configuration.

Default Values

If there are multiple options for an optional property the default value will be labeled `like this`. Other available values will be in `gray`.

Abstract Background



The abstract background engine warps images into animated patterns. There are two styles currently available: `wave1` and `wave2`.

If you don't include a `src` property, a random background will be chosen from the built-in presets each time the content is loaded.

```
abstract-background:
  type: wave1
```

```
abstract-background:
  src: /content/backgrounds/my-preset.json
  type: wave2
  width: 320
  height: 480
```

Properties

`type` `required`

Type: `wave1` | `wave2`

Since: TVS 5.9.0

The style of abstract animation to display.

`src`

Type: `string`

Since: TVS 5.9.0

The URL of the source file to use for the background, relative to the content root or absolute. If empty or unspecified, a random background file will be used.

`width`

Type: `number`

Default: `640`

Since: TVS 5.9.0

The width of the canvas in pixels.

`height`

Type: `number`

Default: `480`

Since: TVS 5.9.0

The height of the canvas in pixels.

Character Generator

The Character Generator engine is loosely based on the Texscan MSI SpectraGen 3 character generator. It is a flexible way to show text and titles and relies on JSON-based configuration files to determine what text to show and how to show it. These config files do not need to be edited by hand however and can be generated using an editor.

Using the Bulletin Editor

Before you can begin using the Character Generator engine you must create a configuration file for it to use. TVS 5.10 and later include this editor in the package.

Opening the editor

Navigate to `http://[your TVS instance]/bulletin-editor`. There is a public version available on the demo site as well.

How the editor works

The editor window exposes two panes:

- **Preview pane:** Shows a preview of what will be displayed in TVS. You can change the resolution of the preview by entering its width and height in the text inputs below the preview as well as scale the preview down to fit it on screen better. You can also drag from the lower right corner of the preview to zoom in / out. There are actions to *Import / Export JSON* configuration files as well as a button to save your current configuration to the browser so that you don't lose it if the page reloads. You can only save one configuration at a time currently; there's a button to create a new project which will reset your browser's stored configuration to a default one.
- **Editor pane:** This is where you can edit what's shown in your configuration. There are tabs for *Header*, *Footer*, and *Pages*.

Project structure

A **Project** is defined as a complete configuration file, including header, footer and pages.

Headers and **footers** are optional and affixed to the top and bottom of the page. They can be edited in their respective tabs and turned off independently using the "Header / Footer enabled" checkboxes.

Projects contain one or more pages of data. The Character Generator engine will cycle between these pages, scrolling long pages with too much text to fit on screen automatically.

Pages are made up of **Lines**, and lines are made up of **Chunks** of text. Each of these components has their own settings.

Pages

Pages can be added and removed using the *Add Page* and *Delete* buttons, as well as reorganized using the *Up* and *Down* buttons. You can duplicate an existing page using the *Duplicate* button. You must have at least one page at all times.

- **Memo:** a note that's only visible in the editor. This is used to identify the page in the editor.
- **Background / foreground colors:** you can override the default colors per-page. For all color pickers you can choose from a palette of colors or specify any color manually.
- **Custom background image:** you can use CSS background shorthand syntax to attach a background image or gradient. Use `url(http://path-to/your-image.jpg)` to use a background image or use a CSS gradient generator to create a background gradient.
- **Font:** you can override the default font with a selection of pixel fonts as well as default browser fonts.
- **Vertical Alignment:** by default pages are aligned to the top of the screen, but you can change this here. Perfect for centered title screens or ads.
- **Transitions:** when the page is shown or hidden you can choose from a variety of fades, wipes and slide transitions and adjust their timing settings in milliseconds.
- **Text Reveal:** the page's lines can be shown sequentially using a number of effects and timings.

Lines

Lines of text can be added using the *Add Line* button and existing lines can be reordered using the *. You can remove

Line Type: there are currently two different types of lines:

- *Text:* rich text which consists of chunks of text which can be formatted differently.
- *Clock:* time and date display which updates automatically.

Depending on the line type, certain options may be unavailable.

- **Font:** applies to all text chunks in this line unless overridden.
- **Alignment:** whether to align text on this line to the left, right or center.
- **Font size:** how big to make the fonts on this line, unless overridden per-chunk.
- **Line height:** spacing between each line. Use to tighten up or loosen spacing within a paragraph.
- **Text width:** stretches or compresses text on the line. Use to fit more text on a line or emphasize text.
- **Padding:** CSS syntax padding for adding custom spacing between lines on the top, right, left and bottom of the line. Usually four values in the format `top left right bottom`. So for instance if you'd like some space on the bottom of a line you might say `0 0 0 1rem` meaning nothing on the top, left, or right but `1rem` on the bottom. For the format you can use `px` for pixels or `rem` for a relative unit based on font size, or any other valid CSS unit.
- **Colors:** the default colors per line; can be overridden per chunk.
- **Character Styles:** customizable shadow and outline effects which can be applied with custom colors. They can be used for style or to make text more visible.
- **Borders:** top and bottom borders can be applied to lines with different colors and styles for each.
- **Crawls:** you can apply a crawl / news ticker effect to a line with adjustable speed. Lower numbers indicate a lower speed and higher numbers will make the text scroll faster.

Chunks

Chunks of text will be displayed on the same line as each other.

- **Text area:** this is where you enter the text that's displayed on the screen. We'll wrap lines automatically unless you're using a Crawl effect on the line.
- **Colors:** you can override background and foreground colors per chunk.
- **Font:** you can specify a font override per chunk.
- **Font size:** you can override font sizes per chunk.
- **Bold, Italic, Underline** effects can be assigned per-chunk.
- **Blink:** text can blink with custom timings in milliseconds (time visible and time hidden)
- **Color cycle:** an animation that can be added to change text color per-letter with custom colors and timing.

Working with Dynamic Data

There are three types of dynamic data available for use in the Character Generator:

- **Variables:** you can turn any chunk into a variable by assigning it a Variable Key. This will allow you to replace data at a later date using TVS YAML or an external JSON file.
- **Block Groups:** you can turn a line or collection of *adjacent* lines into a Block Group by assigning it a Block Key. This allows you to repeat one or more consecutive lines as a group and any variables within will be replaced too.
- **Page Groups:** Repeated pages of data which can contain Block Groups and Variables scoped to the page.

Using the editor, assign a page key (in the ribbon under Pages > General). Assign block keys or variable keys under the "Programmability" tabs for lines and chunks.

Visit the Data tab and you'll see a list of all detected keys. You can add repeating pages and blocks of data here as well as assign local and global variables.

- **Preview in TVS** will open up a draft of the current project including all its data.
- Use the **Export Channel** button to generate YAML that you can paste into your TVS config file.
- Use the **Export Datafile** button to export a JSON file which can be used as-is or as a reference for implementing custom server-side dynamic data.

>
>
> **Caution**
>
> Remember to export your project as well as you need a `src`, not just data!

Using the Character Generator Engine

Once you've exported your project from the Bulletin Editor as a JSON layout template, it's time to put it to use!

```
character-generator:
    src: /content/bulletins/community-board.json
```

```
character-generator:
    src: /content/bulletins/community-board.json
    data: /content/bulletins/community-board-data.json
    refreshInterval: 300
    scrollSpeed: 24
    scrollDelay:
        before: 1
        after: 2
```

```
character-generator:
    src: /content/bulletins/weather-headlines.json
    variables:
        stationName: WGCW
        headline: Community Cleanup Saturday
    lines:
        closings:
        - schoolName: Lincoln High School
            status: Closed
        - schoolName: Washington Elementary
            status: Delayed 2 Hours
    weather:
        provider: weatherapi.com
        apiKey: ${WEATHERAPI_KEY}
        query: 94103
```

Properties

`src` `required`

Type: `string`

Since: TVS 5.10.0

Description: Source URL for the Character Generator layout template file. Create one using the Bulletin Editor and export the project JSON when you're ready to use it in TVS.

`data`

Type: `string`

Since: TVS 5.10.0

Description: URL of a JSON data file containing `variables`, `lines`, and `pages`. This is useful when you want to keep dynamic data separate from the layout template or refresh all dynamic data from a single endpoint.

`refreshInterval`

Type: `number`

Since: TVS 5.10.0

Description: Interval, in seconds, used to refresh JSON-backed dynamic data from `data`, `variables`, `lines`, or `pages` URLs.

`scrollSpeed`

Type: `number`

Since: TVS 5.10.0

Description: Scroll speed in pixels per second for pages that overflow vertically.

`scrollDelay`

Type: `number` | `ICharacterGeneratorScrollDelay`

Since: TVS 5.10.0

Description: Delay, in seconds, before scrolling begins and after the page reaches the end of its scroll. Use a single number to apply the same delay both times, or use the object form below for different values.

`before` `sub-property` `required`

Type: `number`

Since: TVS 5.10.0

Description: Delay, in seconds, before scrolling starts.

`after` `sub-property` `required`

Type: `number`

Since: TVS 5.10.0

Description: Delay, in seconds, after the page reaches the end of its scroll.

`limit`

Type: `number`, 1 or greater

Since: TVS 5.10.0

Description: Number of pages to play before the engine signals that it has finished when nested inside a `loop`. By default, TVS plays all currently loaded pages.

`manualAdvance`

Type: `pages` | `lines` | `chunks`

Since: TVS 5.10.0

Description: Enables manual stepping instead of automatic page progression. Use `Z` to go backward and `X` to go forward. This is useful for live presentations or controlled playback.

`startingPage`

Type: `number` | `random` | `previous` | `next` | `first`

Since: TVS 5.10.0

Description: Controls which page or page group to start on.

- Use a number to select a zero-based page or page group index from the template.
- Use `random` to start on a random page or page group.
- Use `previous` to resume the last page or page group displayed on this channel for this component.
- Use `next` to continue from the page or page group after the last one shown on this channel for this component.
- Use `first` to always start from the first page or page group.

`showBlankBlockGroups`

Type: `boolean`

Default: `false`

Since: TVS 5.10.0

Description: If `true`, block groups with no matching dynamic line data are still shown and their placeholders remain visible.

`showBlankPageGroups`

Type: `boolean`

Default: `false`

Since: TVS 5.10.0

Description: If `true`, page groups with no matching dynamic page data are still shown and their placeholders remain visible.

`variables`

Type: `Record<string, string>` | `string`

Since: TVS 5.10.0

Description: Variable values used to replace bulletin chunks tagged with matching variable keys. You can provide them inline or point to a JSON URL that returns the same object shape.

`lines`

Type: `Record<string, Record<string, string>[]>` | `string`

Since: TVS 5.10.0

Description: Repeating line groups used to duplicate lines tagged with a matching repeat key in the template. Each top-level key is a block group name, and each array entry provides variable values for one repeated copy of that line group.

`pages`

Type: `Record<string, ICharacterGeneratorRepeatingPageEntry[]>` | `string`

Since: TVS 5.10.0

Description: Repeating page groups used to duplicate pages tagged with a matching repeat key in the template. Each top-level key is a page group name, and each array entry represents one generated page.

Dynamic data format

Use `variables` when you only need simple value replacement. Use `lines` when you need to repeat tagged line groups, and use `pages` when you need to repeat entire tagged pages. Each of these properties can be defined inline or loaded from a URL returning JSON.

`variables` sub-property

Type: `Record<string, string>`

Description: Variable overrides applied to a single repeated page entry.

`lines` sub-property

Type: `Record<string, Record<string, string>[]>`

Description: Repeating line groups applied to a single repeated page entry.

`weather`

Type: `IWeatherFramework`

Since: TVS 5.10.0

Description: Optional weather data configuration for bulletin templates that request weather information.

`units` sub-property

Type: `metric` | `imperial`

Default: `metric`

Description: Units used for weather values.

`provider` sub-property

Type: `usa` | `weatherapi.com` | `wttr.in`

Default: `usa`

Description: Weather provider to use.

- `usa`: Uses the built-in USA provider.
- `weatherapi.com`: Uses the WeatherAPI.com provider.
- `wttr.in`: Uses the wttr.in provider.

Since: `weatherapi.com` and `wttr.in` are available in TVS 5.10.0.

`apiKey` `sub-property`

Type: `string`

Since: TVS 5.10.0

Description: API key for the selected weather provider. This is not required for `usa` or `wttr.in`, but it is required for `weatherapi.com`.

`query` `sub-property`

Type: `string`

Since: TVS 5.10.0

Description: Query string for providers that support named or special location lookups. This is supported by `wttr.in` and `weatherapi.com`.

Examples include `London`, `94103`, `@github.com`, and `~Eiffel Tower`.

Color

One of the simplest content engines in Television Simulator is the color engine. It allows you to display a solid color or gradient on the screen, which is useful for placeholders or to mix with other content to create a color filter.

```
color: "#ff0000"
```

```
channels:
  - number: 1
    color: |
      radial-gradient(ellipse at 20% 30%, rgba(56, 189, 248, 0.4) 0%,
transparent 60%),
      radial-gradient(ellipse at 80% 70%, rgba(139, 92, 246, 0.3) 0%,
transparent 70%),
      radial-gradient(ellipse at 60% 20%, rgba(236, 72, 153, 0.25) 0%,
transparent 50%),
      radial-gradient(ellipse at 40% 80%, rgba(34, 197, 94, 0.2) 0%,
transparent 65%)
```

Properties

The color engine has no additional properties; the shorthand `color` is the only property available.

Values can be any value accepted by the CSS `background` property, including solid colors, gradients, and even images. For example:

Gradients:

```
color: "linear-gradient(to right, #ff0000, #0000ff)"
```

Images:

```
color: "url(/content/images/my-image.jpg)"
```

Named colors:

```
color: rebeccapurple
```

Semi-transparent colors:

```
color: "rgba(255, 0, 0, 0.5)"
```

HSL colors:

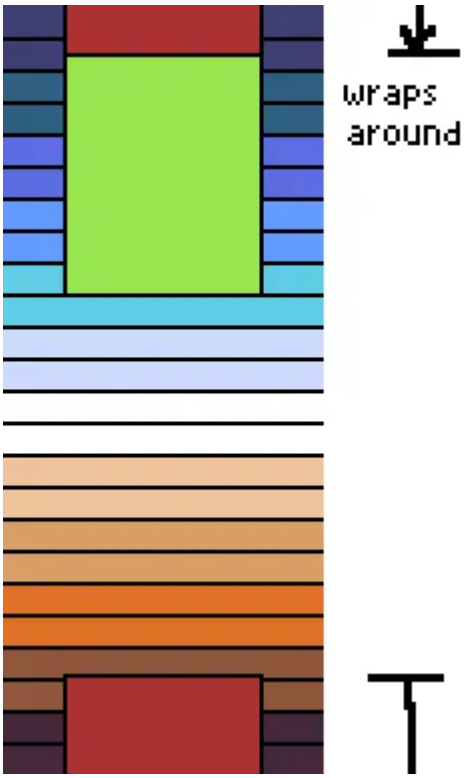
```
color: "hsl(120, 100%, 50%)"
```

Daily Schedule

The Daily Schedule engine allows you to schedule content over a repeating 24-hour period. Similar to the loop engine, you define a list of content that you want to repeat, but here you specify a start and end time (in 24-hour time) for each piece of content, as well as an optional block of content that plays when no other content is playing.

```
- number: 240
  name: Daily Schedule
  abbr: SCHED
  daily-schedule: # The daily schedule engine allows you to define content based on a 24-hour schedule. You can go overnight, but you can't overlap content. The schedule will repeat each day.
    noSignalContent: # Optional. You can specify content to show when there's nothing else scheduled.
      color: black

    content:
    - startTime: "20:00" # Time to start this item, in 24 hour format. Required.
      endTime: "05:00" # Time to end this item, in 24 hour format. Required.
      image: # The content to show. Required.
        src: placeholders/schedule-night.png
        pixelated: true

    - startTime: "05:00"
      endTime: "11:00"
      image:
        src: placeholders/schedule-morning.png
        pixelated: true

    - startTime: "11:00"
      endTime: "20:00"
      image:
        src: placeholders/schedule-day.png
        pixelated: true
```

Wrapping around midnight



If the end time is before the start time, we assume the content plays through the night into the next day. In the diagram above, the red block of content plays into the next day, followed by the green block.

Properties

`content` `shorthand` `required`

Type: `content engine[]`

Since: TVS 5.8.0

Description: List of content to play. Can be any content that TVS can display. When using the shorthand syntax, the list of content is specified directly under the `daily-schedule` key without the `content` property.

> **Heads up!**
>
> Refer to the example code above to learn how to use the sub-properties below.

`startTime` `sub-property` `required`

Type: `string` (in `HH:MM` 24-hour format)

Description: The time to start showing this content, in 24-hour format.

`endTime` `sub-property` `required`

Type: `string` (in `HH:MM` 24-hour format)

Description: The time to stop showing this content, in 24-hour format.

`noSignalContent`

Type: `content engine`

Since: TVS 5.8.0

Description: Content to play when nothing else is on the air. Optional.

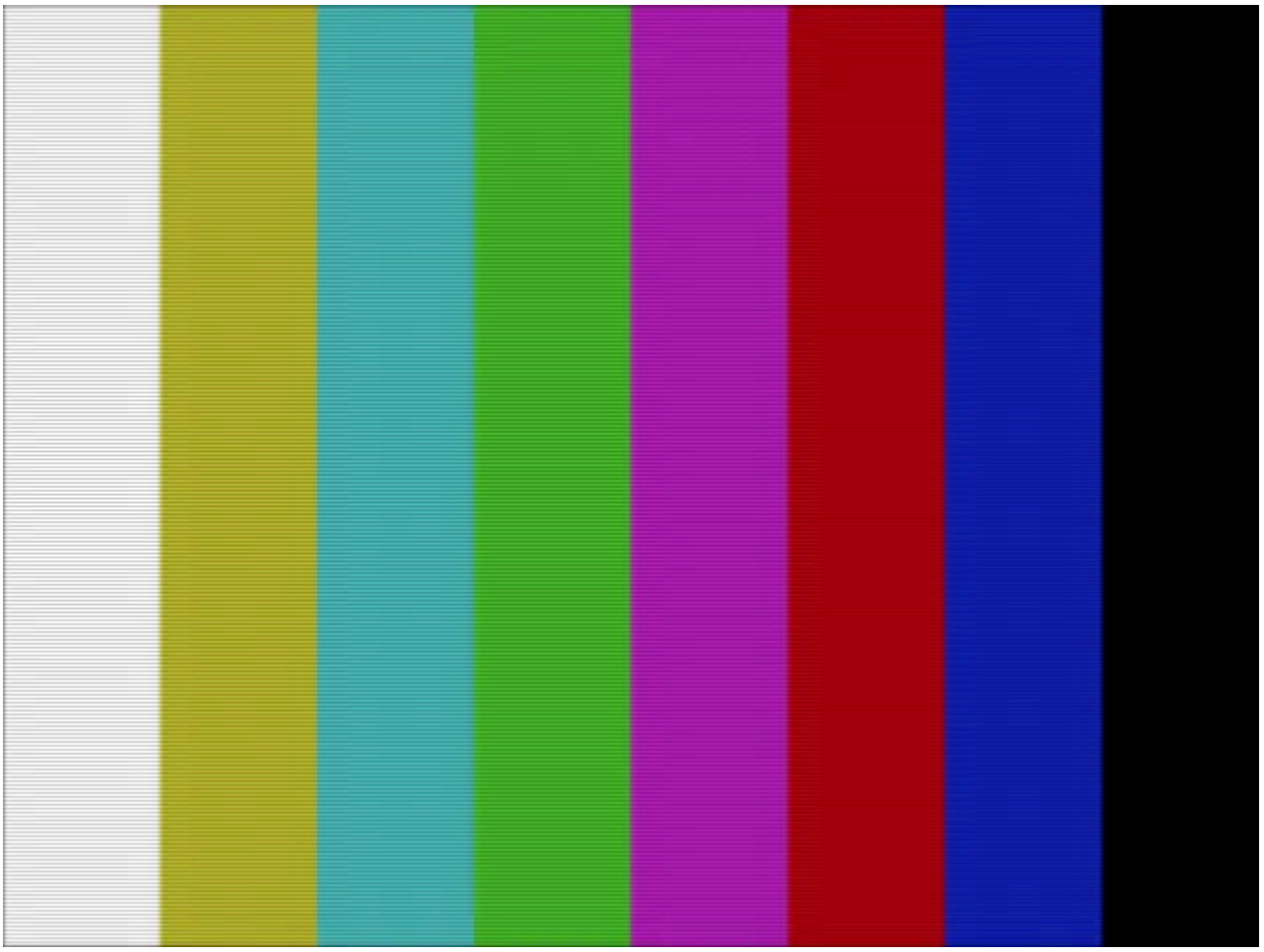
Generator

Test Pattern

```
generator: test-pattern
```

Generates a SMPTE color bars test pattern with a 1000hz tone.

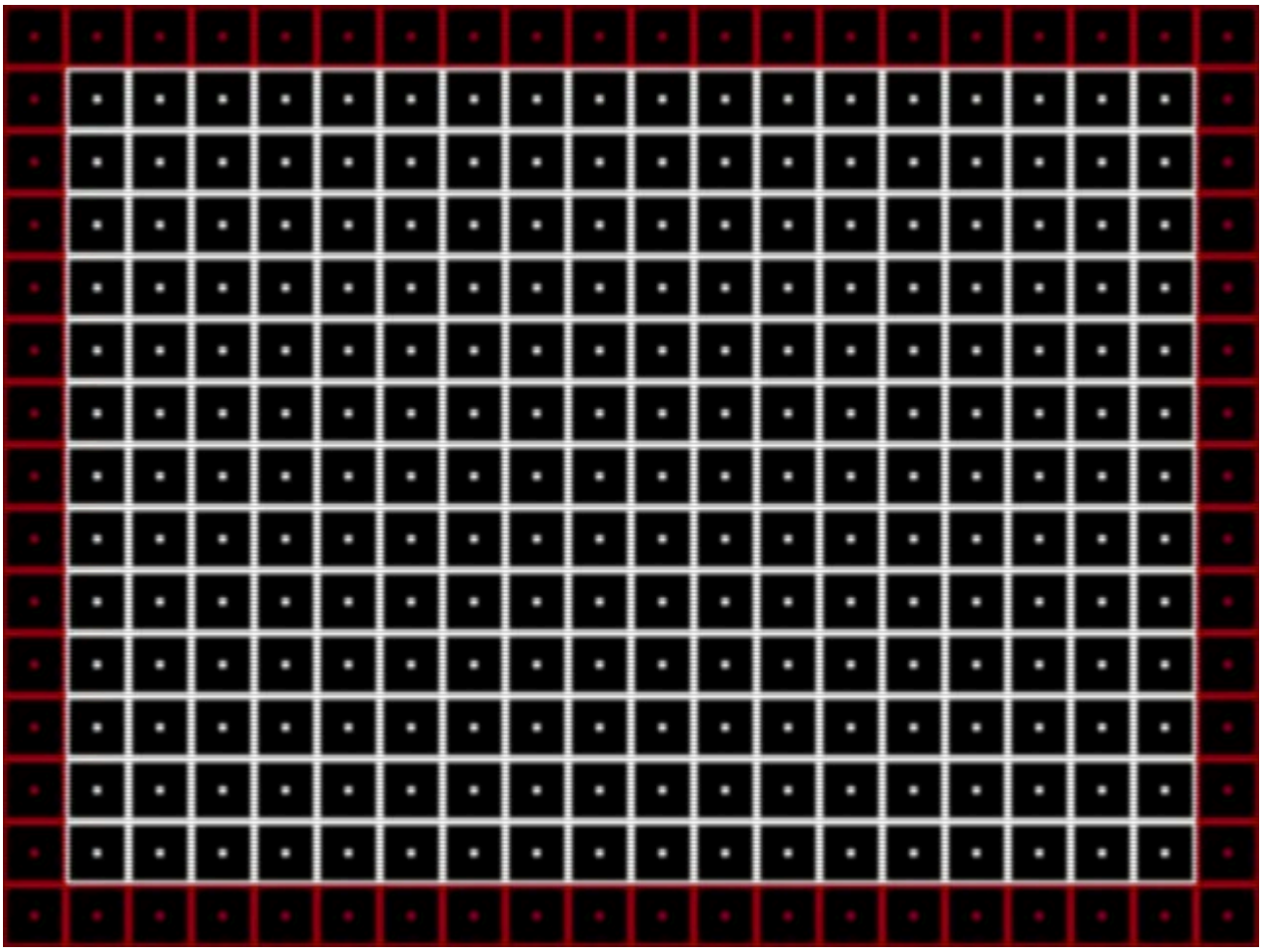
Color Bars



```
generator: color-bars
```

Generates EBU colour bars test pattern with a 1000hz tone.

Grid



```
generator: grid
```

Generates a CRT alignment grid with a 1000hz tone.

Properties

`type` shorthand required

Type: `test-pattern` | `color-bars` | `grid`

Description: The type of generator to use.

`volume`

Type: `number` | `false`

Description: The volume level of the tone, ranging from `0.0` (muted) to `1.0` (full volume). The default value is `1`.

IFrame

The `iframe` engine allows you to embed external web content directly into your Television Simulator channels. This can be useful for displaying web pages, web apps, or other online content.

Properties

`src` `shorthand` `required`

Type: `string`

The URL of the web page to embed in the iframe. This should be a fully qualified URL, including the `http://` or `https://` protocol.

`title`

Type: `string`

Description: The title of the iframe for accessibility purposes. The default value is `Embedded content`.

`refreshIntervalSeconds`

Type: `number`

Description: Number of seconds before automatically refreshing the iframe. Set to 0 to disable automatic refreshing. The default value is `0`.

`allowInteraction`

Type: `boolean`

Description: Whether to allow user interaction with the iframe content (clicking, scrolling, etc.). The default value is `false`.

`options`

Type: `object`

Since: TVS 5.6.0

Description: An optional object containing key-value pairs that can be passed to the embedded application via the TVS IFrame API. See the TVS IFrame API documentation for more details on how to use this feature.

Images

Television Simulator can display any image your browser can render.

```
image: /content/images/my-image.jpg
```

```
image:
  src: /content/images/my-image.jpg
  aspectRatioBehavior: cover
```

Properties

`src` shorthand required

Type: `string`

The URL of the image to display, relative to the content root or absolute.

`aspectRatioBehavior`

Type: `cover` | `contain` | `stretch` | `repeat` | `blur`

How the image should be displayed in relation to the screen size. `contain` will fit the image inside the screen, `cover` will fill the screen while cropping the image, `stretch` will stretch the image to fill the screen, `repeat` will tile the image, and `blur` will apply a blur effect to edges of the image.

`pixelated`

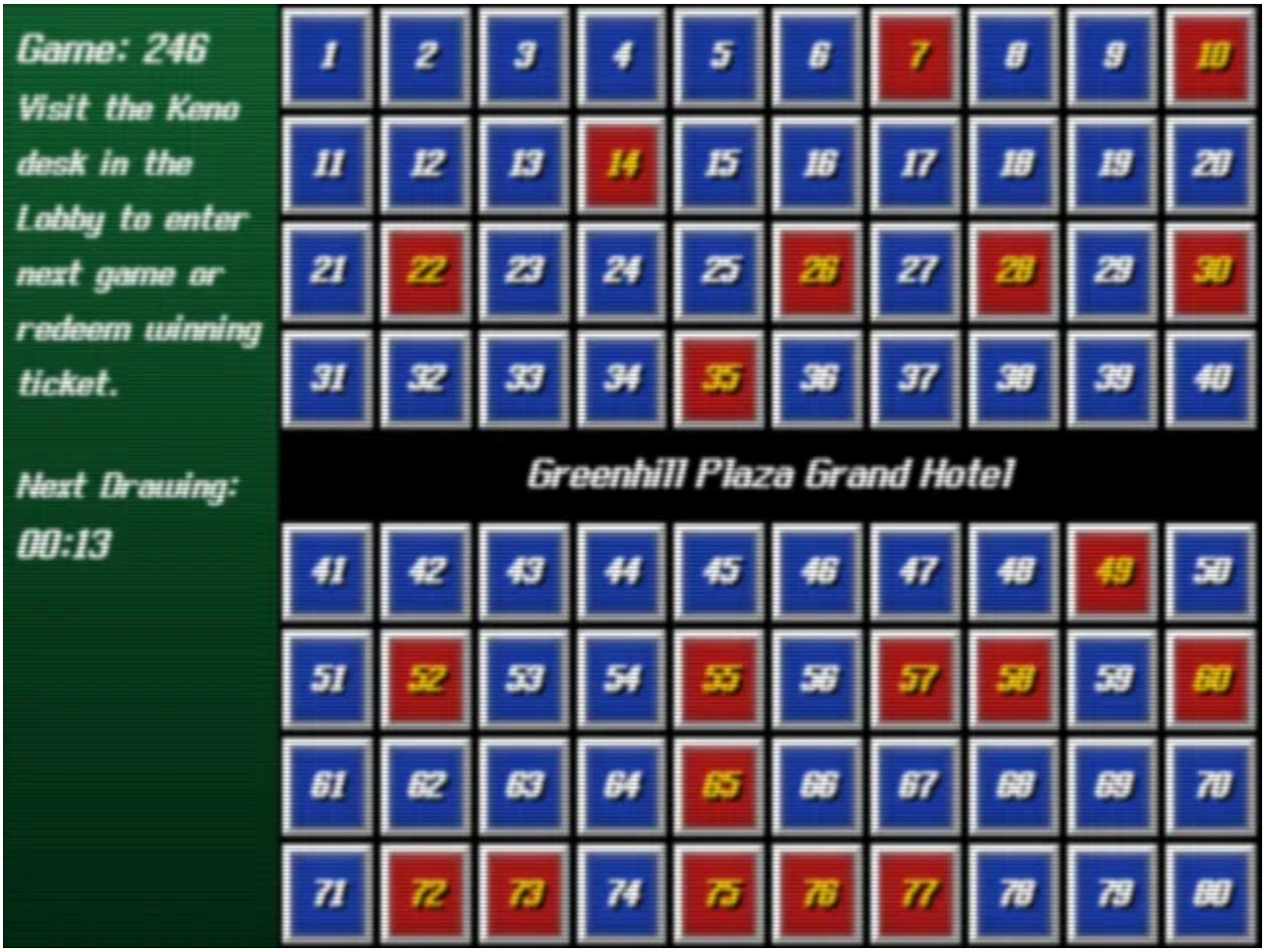
Type: `boolean`

Since: TVS 5.8.0

Description: Whether to display the image with a pixelated effect. This can be useful for low-resolution images or pixel art. The default value is `false`.

Keno

Shows a keno lottery game simulation. The look and feel of this engine is modeled from its appearance on CCTV from casino hotels, where guests can check the results of a game from their room. 20 numbers are chosen randomly from a pool of 80.


Game: 246
Visit the Keno desk in the Lobby to enter next game or redeem winning ticket.
Next Drawing:
00:13
Greenhill Plaza Grand Hotel

```
keno:
  drawingMinutes: 1
  drawIntervalMs: 600
  centerText: Greenhill Plaza Grand Hotel
```

Properties

`drawingMinutes`

Type: `number`

Default: `5`

The number of minutes between each keno drawing, starting from 12 AM.

`drawIntervalMs`

Type: `number`
Default: `1000`

The number of milliseconds between each ball being drawn.

`volume`

Type: `number` | `false`
Default: `1`

Volume of the sound effects, from `0` (muted) to `1` (full volume). Set to `false` to disable sound effects.

`centerText`

Type: `string`

The text to display in between the halves of the screen.

Loop

The Loop engine shows other content in a continuous loop. This is a fundamental component that can be used to create more sophisticated channel content by nesting loops, using references and shuffling.

```
loop:
- image: picture.png
  duration: 3
- video: video.mp4
- color: red
  duration: 2
```

```
loop:
  shuffle: cycle
  content:
  - image: picture.png
    duration: 3
  - video: video.mp4
  - color: red
    duration: 2
```

Properties

`content` `shorthand` `required`

Type: `content engine[]`

Description: List of content to loop. Can be any content that TVS can display. When using the shorthand syntax, the list of content is specified directly under the `loop` key without the `content` property.

>
>
> **Heads up!**
>
> Refer to the example code above to learn how to use the `duration` sub-property.

`duration` `sub-property`

Type: `number`

Description: Content inside a loop has a special optional property called `duration`, which specifies how long (in seconds) to display that content before moving to the next item in the loop. If `duration` is not specified for an item, the default behavior is to display it for its full length (e.g., the full duration of a video).

Not all content has “end detection”; in general, video content does but images and other content does not. If you don’t specify a duration for content without end detection, the duration defaults to 30 seconds.

`shuffle`

Type: `random` | `cycle` | `false`

Added in version: `5.0.0`

Whether to shuffle the content in the loop. If set to `random`, the content will be displayed in a random order. If set to `cycle`, the content will be displayed in a random order but will ensure that all items are shown once before any item is repeated. If set to `false`, the content will be displayed in the order it is defined.

`limit`

Type: `number`, 1 or greater

Added in version: `5.5.0`

Limits the number of items from the content list to include in the loop.

By default we don’t apply a limit. If set to a number less than the total number of items, only that many items will be randomly selected from the content list to be included in the loop.

This is useful when combined with `shuffle` to create a pool of content to choose from; for example, using `shuffle: random` and `limit: 1` will grab one random piece of content whenever the loop is displayed. Nest this loop inside another loop and now you’ve created, for instance, an ad pool that will display a random ad from the loop

each time it’s shown. Say you make that a `ref` then you can reuse it whenever you’d like to show an ad!

Refs (Referenced Content)

Content can be defined as a reference and reused in multiple places throughout your Television Simulator config file. You can override top-level properties of referenced content at the time that you use it.

`refs` section

In the TVS configuration file you can define reusable content by adding a `refs` section at the top level of the file.

Keys in the `refs` section are arbitrary names that you can use to reference the content later in the file.

```
refs:
  movie: # The reference "key". You can name this anything, but it must be unique, start with a letter, and only use letters, numbers, hyphens, and underscores.
    video: my-movie.mp4

  network-logo:
    image: images/logo.png
```

In the example above we've defined two pieces of reusable content: `movie` and `network-logo`. To use them, you'll use the `ref` engine in a channel. So instead of:

```
channels:
- number: 1
  abbr: MOV
  video: my-movie.mp4
```

You would do:

```
channels:
- number: 1
  abbr: MOV
+  ref: movie
```

And TVS will replace the `ref: movie` line with the content defined in the `refs` section.

Overriding properties

Properties in references can be overridden when you use them.

>
>
> **Caution**
>
> To override properties, you can't use shorthand syntax when defining the reference.
>
> ```
> refs:
>   non-overridable-reference:
>     video: my-video.mp4 # Cannot be overridden because shorthand syntax is used.
>
>   overridable-reference:
>     video:
>       src: my-video.mp4 # Can be overridden because full syntax is used.
> ```

To override properties, you'll use the `ref` engine in a slightly different way:

```
refs:
  movie:
    video:
      src: my-movie.mp4

channels:
- number: 1
  abbr: MOV
+  ref:
+    name: movie
+    overrides:
+      volume: 0
```

You'll specify a `name` for the reference key you want to use, then define `overrides` with the properties you want to change.

The example above will render like this:

```
video:
  src: my-movie.mp4
  volume: 0
```

Nested references

You can nest references within other references for more complex reuse scenarios.

```
refs:
  ad1:
    image:
      src: placeholders/example-ad.webp

  ad2:
    mix:
      a:
+       ref: ad1
        mixMode: difference
      b:
        ref: intro
        mixMode: exclusion
```

In the example above, the `ad2` reference uses the `ad1` reference within a `mix` layout.

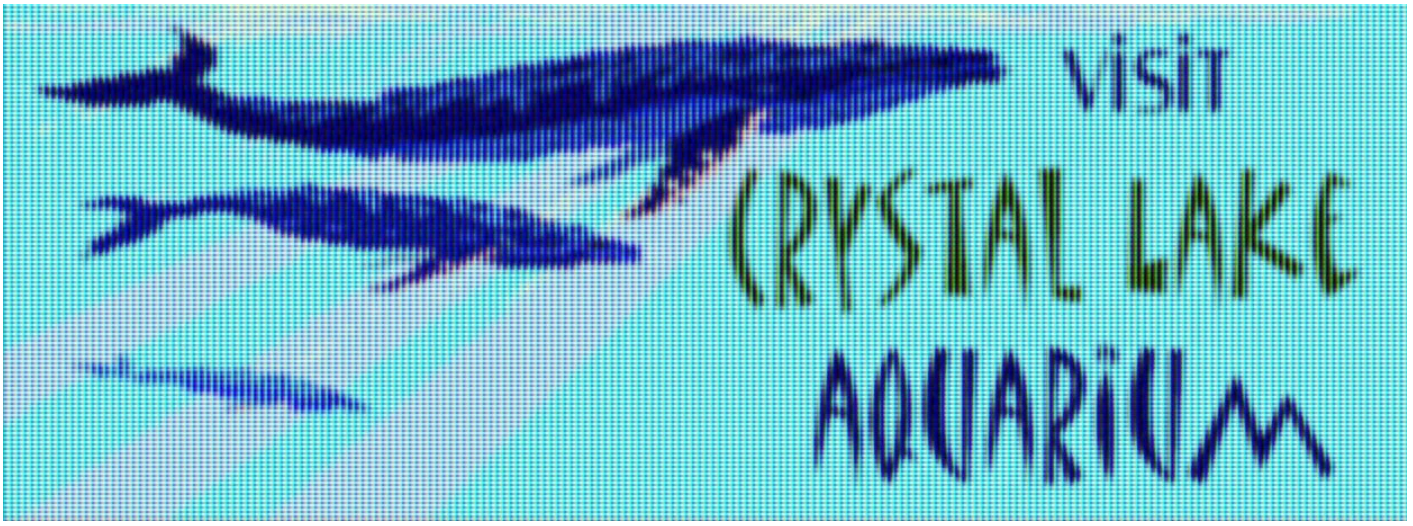
Danger

Beware of circular references! If a reference points to another reference that eventually points back to the original reference, it will create an infinite loop and likely crash TVS.

TVS does not currently detect circular references, so it’s up to you to avoid them.

```
# **Do not do this, and do not do anything like this!**
refs:
  ref-a:
-   ref: ref-b

  ref-b:
-   ref: ref-a
```

Slideshow



Slideshows will show a series of images in sequence. If you use a slideshow inside of a `loop` component, the loop will advance to the next content after the slideshow has played through all the images a single time.

```
slideshow:
  duration: 10
  images:
  - placeholders/guide-ad.png
  - placeholders/tips-01.png
  - placeholders/guide-ad-2.png
```

Properties

`images` `required`

Type: `string[]`

The URLs of the images to display, relative to the content root or absolute.

`duration`

Type: `number`
Default: `10`

The duration of each slide in seconds.

Video

Television Simulator can display video compatible with your browser. This includes common formats such as MP4 and WebM, as well as HTTP Live Streaming (HLS) and DASH. TVS also supports remote video playback from YouTube. It uses Vidstack under the hood to manage playback.

```
video: /path/to/video.mp4
```

```
video: https://www.youtube.com/watch?v=Bb0dJxCg_sw
```

```
video:
  src: https://www.example.com/path/to/livestream.m3u8
  provider: hls
  volume: 0.85
```

Properties

`src` `shorthand` `required`

Type: `string`

Description: The URL of the video file to play. This can be a local file (hosted through web server) or a remote URL.

`provider`

Type: `video` | `youtube` | `hls` | `dash`

Description: The video provider to use for playback. If not specified, TVS will attempt to determine the provider based on the URL.

`volume`

Type: `number` | `false`

Description: The volume level to use for playback, ranging from `0.0` (muted) to `1.0` (full volume). The default value is `1`.

`aspectRatioBehavior`

Type: `contain` | `cover` | `stretch`

Since: TVS 5.7.0

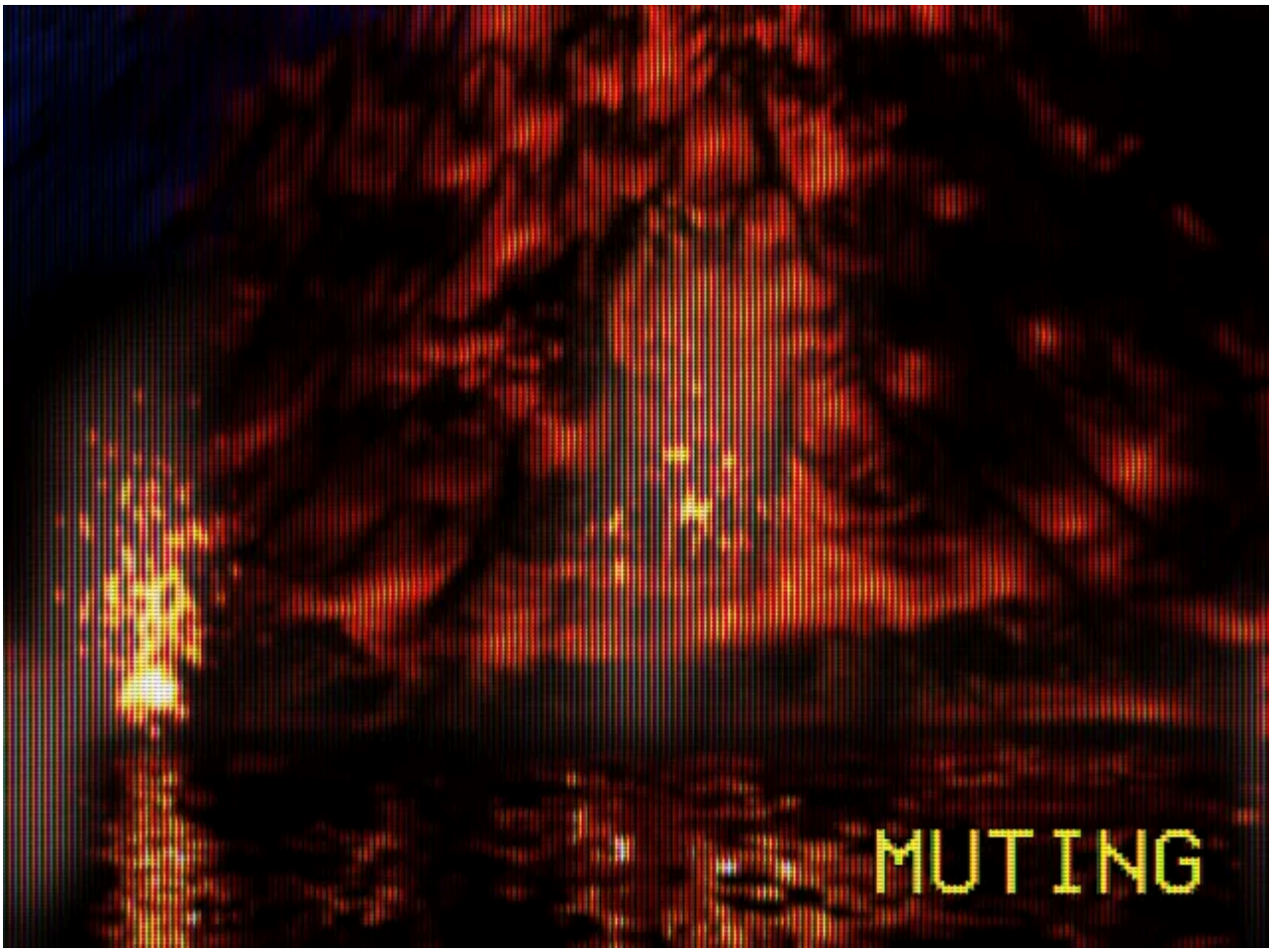
Description: Determines how the video should fit within its container while maintaining its aspect ratio.

- `contain`: The video will be scaled to fit within the container while preserving its aspect ratio. This may result in letterboxing or pillarboxing.
- `cover`: (Pan and scan) The video will be scaled to cover the entire container while preserving its aspect ratio. This may result in cropping.
- `stretch`: The video will be stretched to fill the entire container, ignoring its aspect ratio. This may result in distortion.

Visualizer

The visualizer engine shows one of 100 really cool visualizations that react to the audio being played. Under the hood we're using the Butterchurn library, which itself is a WebGL-based version of the legendary Milkdrop library to render these visualizations.

```
visualizer: https://example.com/audio-stream.mp3
```




```
visualizer:
  src: https://example.com/audio-stream.mp3
  visualization: 27
  visualizationDuration: 20
  visualizationBlendDuration: 1
```

Properties

src shorthand required

Type: `string`

The URL of the audio stream to visualize.

---

visualization

Type: `number`

Default: `0`

The ID of the visualization to use. Leave empty or set to 0 for a random one.

`visualizationDuration`

Type: `number`

Default: `30`

How many seconds before choosing a new random visualization. Use 0 to stay on the same visualization indefinitely.

`visualizationBlendDuration`

Type: `number`

Default: `2`

How many seconds to blend visualizations together when switching.

Weather

The Weather engine displays current weather conditions for a location in the style of the WeatherSTAR 4000. It can use automatic location lookup when available, fixed coordinates, or provider-specific location queries.

```
weather:
    provider: usa
    units: imperial
  latitude: 40.7128
  longitude: -74.0060
```

```
weather:
  provider: wttr.in
  query: London
  units: metric
```

```
weather:
  provider: weatherapi.com
  apiKey: your_api_key_here
  query: 94103
  logo: /content/images/my-station-logo.png
```

Different Data Providers

The available weather data and features may vary based on the selected provider. For example, some providers may support named location queries while others require GPS coordinates.

Properties

`latitude`

Type: `number`

Description: Latitude of the location to fetch weather for. If not provided, TVS will try to determine the location automatically when possible.

`longitude`

Type: `number`

Description: Longitude of the location to fetch weather for. If not provided, TVS will try to determine the location automatically when possible.

If automatic lookup fails, see the Weather Location Error troubleshooting page.

`logo`

Type: `string`

Description: URL of an optional logo to display in the upper-left corner of the weather screen.

units

Type: `metric` | `imperial`

Default: `metric`

Description: Units used for weather values.

---

provider

Type: `usa` | `weatherapi.com` | `wttr.in`

Default: `usa`

Description: Weather provider to use.

- `usa`: Uses the built-in USA provider. Available since TVS 2.0.0.
- `weatherapi.com`: Uses the WeatherAPI.com provider. Available since TVS 5.10.0.
- `wttr.in`: Uses the wttr.in provider. Available since TVS 5.10.0.

---

apiKey

Type: `string`

Since: TVS 5.10.0

Description: API key for the selected weather provider. This is not required for `usa` or `wttr.in`, but it is required for `weatherapi.com`.

---

query

Type: `string`

Since: TVS 5.10.0

Description: Query string for providers that support named or special location lookups. This is supported by `wttr.in` and `weatherapi.com`.

Examples include `London`, `94103`, `@github.com`, and `~Eiffel Tower`.

Youtube Playlist

Television Simulator can play YouTube playlists in order or shuffled.

```
youtube-playlist: PLU1WjbywtZxEGdTPm-qB0KzPGkRPw6npl
```

```
youtube-playlist:
  src: PLU1WjbywtZxEGdTPm-qB0KzPGkRPw6npl
  continueFromLast: false
  shuffle: true
```

Properties

`src` shorthand required

Type: `string`

The ID of the YouTube playlist to play. This can be found in the URL of the playlist, for example, `https://www.youtube.com/playlist?list=PL1234567890abcdef` would have an ID of `PL1234567890abcdef`.

`startingVideo`

Type: `number`

The index of the video to start playback from. This is zero-based, so the first video in the playlist would be `0`, the second video would be `1`, and so on. If not specified, playback will start from the first video in the playlist.

`shuffle`

Type: `true` | `false`

Whether to shuffle the playlist before playback. If set to `true`, the videos will be played in a random order. If not specified, playback will be in the order the videos appear in the playlist.

`loop`

Type: `true` | `false`

If set to `true`, the playlist will loop back to the beginning after the last video has finished playing. If set to `false`, playback will stop after the last video in the playlist.

> **Caution**
>
> If you're using a YouTube playlist inside of a `loop` engine, you must set the `loop` property to `false` to avoid preventing the loop engine from advancing to the next item.

`continueFromLast`

Type: `true` | `false`

If set to `true`, the playlist will continue playback from the last video that was played when the engine is restarted. If set to `false`, playback will start from the beginning of the playlist or a random video if `shuffle` is enabled. This is useful for resuming playback after a channel has been stopped or restarted.

`volume`

Type: `number`

The volume level to play the videos at, ranging from `0.0` (muted) to `1.0` (maximum volume). If not specified, the default volume level will be used.

Layout Primer

Layouts are ways to arrange multiple different pieces of content on the screen at the same time. Currently there are a few primitive layouts available, but there are plans to rework them to be more flexible in the future.

”A” and “B” components

The layouts that are avilable today support two abstract components, “A” and “B”. These components can be any engine, including another layout. This allows for nesting layouts inside of other layouts to create more complex arrangements. Let’s look at an example:

```
h-split:
  a:
    image: placeholders/example-ad.webp
  b:
    generator: color-bars
```

This layout uses the `h-split` layout to arrange two components side-by-side. The “A” component is an image engine that displays an example ad, and the “B” component is a generator engine that displays color bars. The result is a screen that shows the ad on the left and the color bars on the right. As a general rule, “A” is the first or primary component, and “B” is the secondary component.

Horizontal Split

Horizontal split (`h-split`) is a layout that divides the screen into two sections, side by side. The left section is referred to as “A” and the right section as “B”. Each section can contain any engine, including another layout, allowing for complex arrangements of content.

```
h-split:
  a:
    image:
        src: placeholders/example-ad.webp
        aspectRatioBehavior: contain
  b:
    generator: color-bars
```

This layout will display the example ad on the left side of the screen and color bars on the right side.

Properties

a `required`

Type: `content engine`

The content to display in the left section of the screen. This can be any engine, including another layout.

---

b `required`

Type: `content engine`

The content to display in the right section of the screen. This can be any engine, including another layout.

`splitPercentage`

Type: `number`

Description: The percentage of the screen width to allocate to the left section (A), ranging from `0.0` to `1.0`. For example, `0.3` would make the left section 30% of the screen width and the right section 70%. The default value is `0.5`.

`gapSize`

Type: `number`

Description: The pixel size of the gap between the left and right sections. The default value is `0`.

`gapColor`

Type: `string`

Description: The color of the gap between the left and right sections. Accepts any valid CSS color value. The default value is `black`.

Vertical Split

Vertical split (`v-split`) is a layout that divides the screen into two sections, one on top of the other. The top section is referred to as “A” and the bottom section as “B”. Each section can contain any engine, including another layout, allowing for complex arrangements of content.

```
v-split:
  a:
    image: /placeholders/cable-ad-2.png
  b:
    guide:
    theme: "1989"
    themeOptions:
      ticker:
      - "Hypercom cable - connecting you to the future"
      - "Pay-Per-View Movies - Ch 1 - Straight from Hollywood to your home,
anytime!"
```

This layout will display an image on the top half of the screen and a program guide on the bottom half.

Properties

a `required`

Type: `content engine`

The content to display in the top section of the screen. This can be any engine, including another layout.

---

b `required`

Type: `content engine`

The content to display in the bottom section of the screen. This can be any engine, including another layout.

`splitPercentage`

Type: `number`

Description: The percentage of the screen height to allocate to the top section (A), ranging from `0.0` to `1.0`. For example, `0.3` would make the top section 30% of the screen height and the bottom section 70%. The default value is `0.5`.

`gapSize`

Type: `number`

Description: The pixel size of the gap between the top and bottom sections. The default value is `0`.

`gapColor`

Type: `string`

Description: The color of the gap between the top and bottom sections. Accepts any valid CSS color value. The default value is `black`.

Mix

The `mix` layout allows you to overlay one piece of content on top of another. “A” is in the background and “B” is in the foreground. You can specify opacity and mix mode for each layer to create interesting effects.

```
mix:
  a:
    opacity: 0.6
    image: placeholders/example-program.webp
  b:
    mixMode: difference
    generator: test-pattern
```

Properties

a `required`

Type: `content engine`

The content to display in the left section of the screen. This can be any engine, including another layout.

---

b `required`

Type: `content engine`

The content to display in the right section of the screen. This can be any engine, including another layout.

Layer Properties

>
> **Tip**
>
> The following properties can be specified for both layer A and layer B.

`mixMode`

Type: `normal` | `multiply` | `screen` | `overlay` | `darken` | `lighten` | `color-dodge` | `color-burn` | `hard-light` | `soft-light` | `difference` | `exclusion` | `hue` | `saturation` | `color` | `luminosity` | `plus-darker` | `plus-lighter`

The blend mode to use with each layer. Accepts any valid CSS `mix-blend-mode` value.

`opacity`

Type: `number`

The opacity of the layer, ranging from `0.0` (fully transparent) to `1.0` (fully opaque). The default value is `1.0`.

Picture-in-Picture

Picture-in-Picture (`pip`) is a layout that displays a main content area with a smaller overlay window positioned anywhere on the screen. The main content fills the entire screen while the overlay content appears in a smaller window on top.

```
pip:
  a:
    image: placeholders/example-program.webp
  b:
    generator: test-pattern
```

This layout will display color bars as the main content with a YouTube video overlay positioned in the bottom-right corner.

Properties

a `required`

Type: `content engine`

Description: The content to display as the main background that fills the entire screen. This can be any engine, including another layout.

---

b `required`

Type: `content engine`

Description: The content to display in the picture-in-picture overlay window. This can be any engine, including another layout.

---

background

Type: `string`

Since: TVS 5.7.0

Description: The background color to display behind the picture-in-picture content if it does not cover the entire overlay window. Accepts any valid CSS color value. The default value is `transparent`.

---

borderSize

Type: `number` | `string`

Description: The border size around the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). The default value is `0`.

---

borderColor

Type: `string`

Description: The color of the border around the picture-in-picture content. Accepts any valid CSS color value. The default value is `black`.

`width`

Type: `number` | `string`

Description: The width of the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If not specified, it defaults to 1/4 of the screen width.

`height`

Type: `number` | `string`

Description: The height of the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If not specified, it defaults to 1/4 of the screen height.

`top`

Type: `number` | `string`

Description: Distance from the top edge of the screen to position the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If none of the position options (`top`, `left`, `right`, `bottom`) are specified, the overlay defaults to bottom-right positioning.

`left`

Type: `number` | `string`

Description: Distance from the left edge of the screen to position the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If none of the position options (`top`, `left`, `right`, `bottom`) are specified, the overlay defaults to bottom-right positioning.

`right`

Type: `number` | `string`

Description: Distance from the right edge of the screen to position the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If none of the position options (`top`, `left`, `right`, `bottom`) are specified, the overlay defaults to bottom-right positioning.

`bottom`

Type: `number` | `string`

Description: Distance from the bottom edge of the screen to position the picture-in-picture content. Can be specified in pixels (as a number) or any valid CSS size unit (as a string). If none of the position options (`top`, `left`, `right`, `bottom`) are specified, the overlay defaults to bottom-right positioning.

Launcher Changelog

1.1.0 (2026-05-24)

- Added support for macOS. The launcher is now available for Windows, Linux and macOS.
- Added ability to log into the Greenhill Software Center supporters site to get prerelease and testing versions of TVS.
- Fixed issues where the launcher's tray sub-menus didn't always populate by moving the version selector and configuration pickers to a window with a list box instead.

1.0.0-preview2 (2025-11-23)

- Added cross-platform support for Linux
- Found a better tray icon library that supports dropdown menus and doesn't require CGO
- Added a CLI mode for Linux users without a GUI; Windows users can use this version using `tvs-launcher-cli.exe`

The Linux version installs to `~/.local/share/tvs-launcher`.

1.0.0-preview (2025-11-21)

Initial prerelease of Television Simulator Launcher for Windows.

Features

- Download and install latest version of Television Simulator directly from GitHub releases.
- Switch between multiple releases and configurations.
- Integrated HTTP server that serves TVS; no need for a separate web server.
- Runs in the background as a system tray application.

Installation

Put `tvs-launcher.exe` in a folder where you want to run the launcher from.

When you run it for the first time, it will create the necessary folders and be ready to download the latest version of Television Simulator. Click on the tray icon to open the launcher.

Usage

First, update to the latest version of TVS using the Update command at the top of the menu. Once updated, the server will start automatically, by default at http://localhost:3990

You can change the server port and host in the `settings.json` file located in the root folder.

Configuration Management

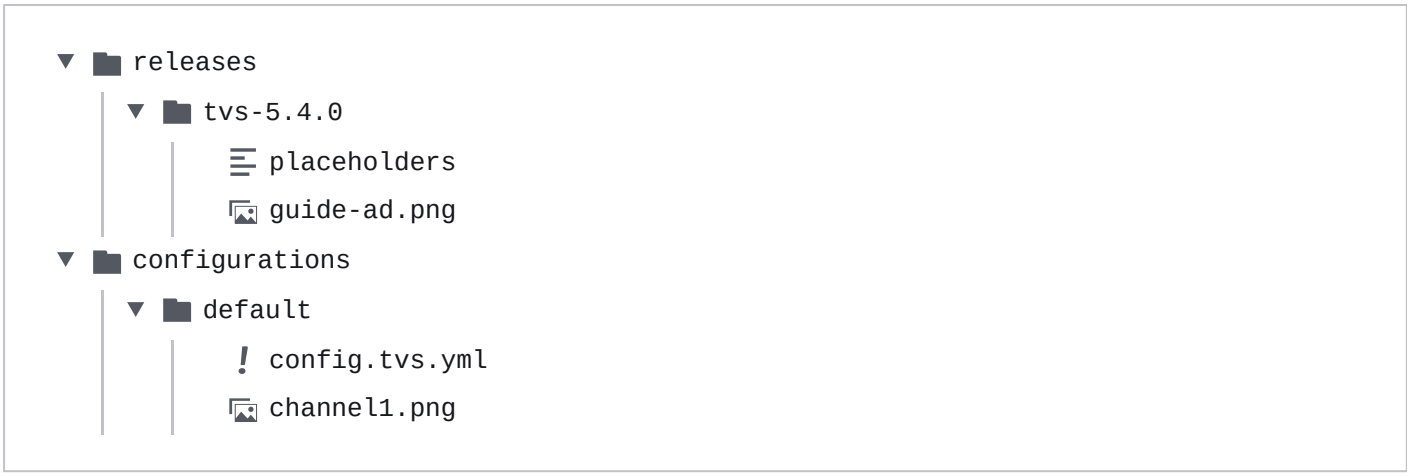
All of your user data and configurations are stored in the `configurations` folder. When updating to a new version, your existing data will be preserved. New installations will start with the latest demo config file.

You can create multiple configurations and switch between them using the launcher; only one will be served at a time.

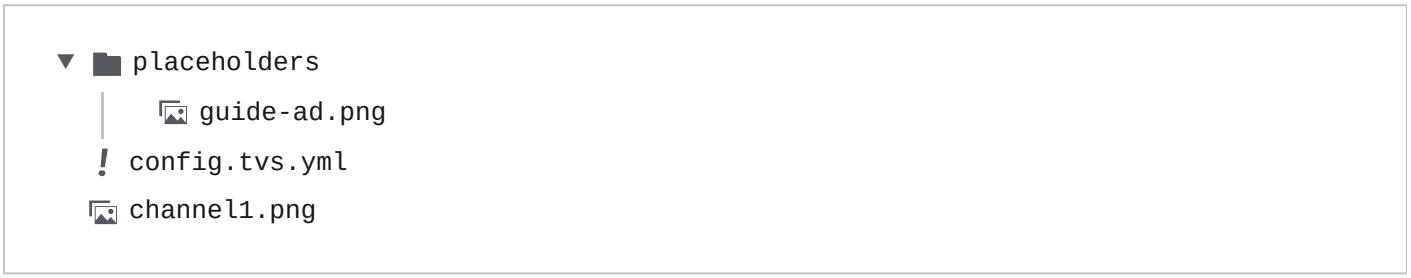
The way that the web server serves files is based on merging your configuration folder with the installed release. For instance if you have a config file like this:

```
channels:
  - number: 1
    image: channel1.png

  - number: 2
    image: placeholders/guide-ad.png
```

And on your disk you've got:



The server will treat it as if the files exist in the same folder:



Using the Launcher

Using the new Television Simulator Launcher is the easiest way to install, update and manage TVS on Windows and Linux. It runs in the system tray and includes an integrated web server that serves TVS without the need for anything else. In addition, the launcher will manage your configurations and keep your user data separate when updating to a new release.

Installation

TVS Launcher is available currently for Windows, Linux and macOS. Download the latest version for your platform from the GitHub releases page.

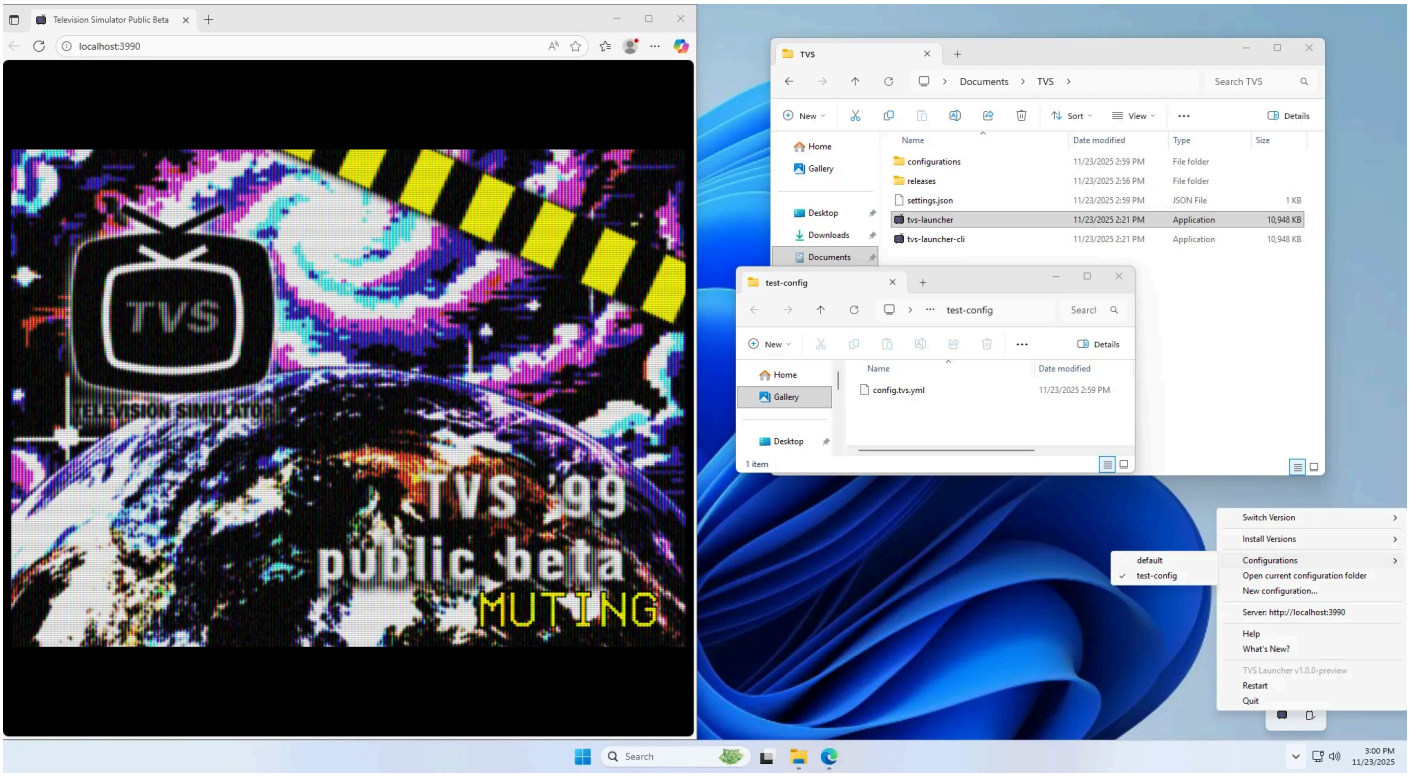
Download Launcher for Windows

Download Launcher for macOS

Download Launcher for Raspberry Pi

Download Launcher for other Linux distributions on GitHub

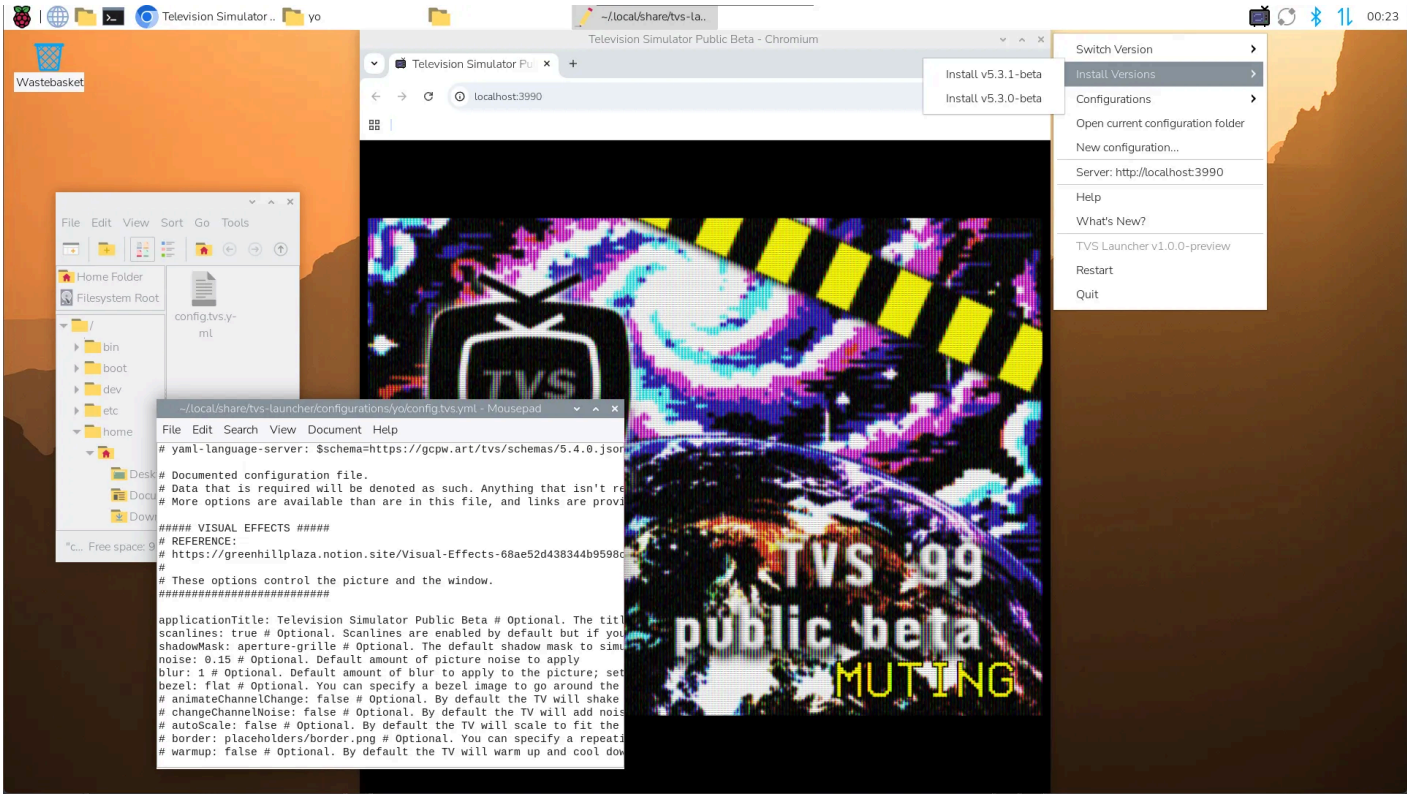
Windows



1. Put `tvs-launcher.exe` in a folder where you want to run the launcher from.

2. When you run it for the first time, it will create the necessary folders and be ready to use.

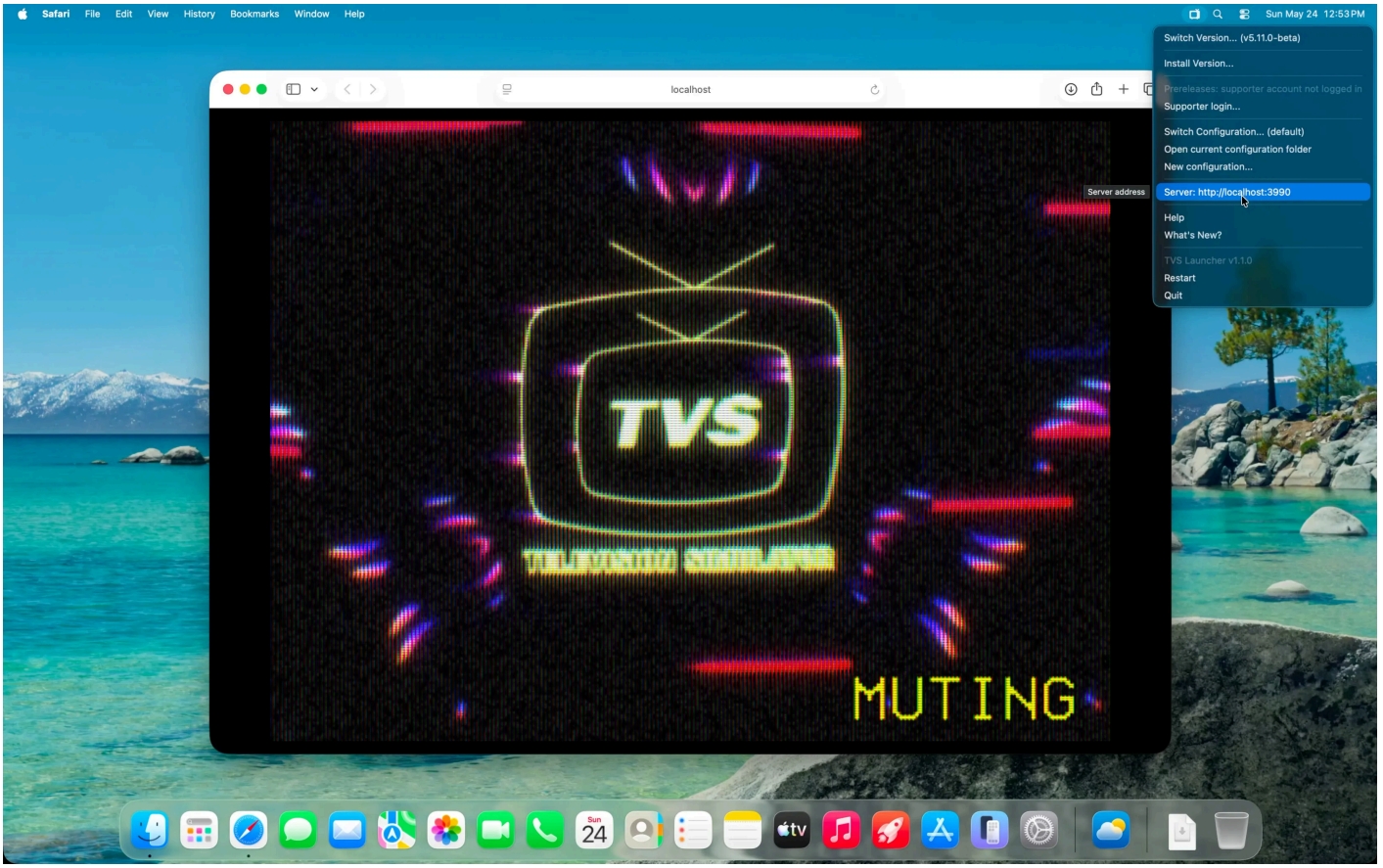
Linux


Television Simulator ..
yo
~/.local/share/tvs-la..
00:23
Television Simulator Public Beta - Chromium
Wastebasket
Switch Version
Install Versions
Configurations
Open current configuration folder
New configuration...
Server: http://localhost:3990
Help
What's New?
TVS Launcher v1.0.0-preview
Restart
Quit
Install v5.3.1-beta
Install v5.3.0-beta
localhost:3990
File Edit View Sort Go Tools
Home Folder
Filesystem Root
config.tvs.y-ml
bin
boot
dev
etc
home
~/.local/share/tvs-launcher/configurations/yo/config.tvs.yml - Mousepad
File Edit Search View Document Help
yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.4.0.json
Documented configuration file.
Data that is required will be denoted as such. Anything that isn't re
More options are available than are in this file, and links are provi
VISUAL EFFECTS
REFERENCE:
https://greenhillplaza.notion.site/Visual-Effects-68ae52d438344b9598c
#
These options control the picture and the window.
############################
applicationTitle: Television Simulator Public Beta # Optional. The titl
scanlines: true # Optional. Scanlines are enabled by default but if you
shadowMask: aperture-grille # Optional. The default shadow mask to simu
noise: 0.15 # Optional. Default amount of picture noise to apply
blur: 1 # Optional. Default amount of blur to apply to the picture; set
bezel: flat # Optional. You can specify a bezel image to go around the
animateChannelChange: false # Optional. By default the TV will shake
changeChannelNoise: false # Optional. By default the TV will add nois
autoScale: false # Optional. By default the TV will scale to fit the
border: placeholders/border.png # Optional. You can specify a repeati
warmup: false # Optional. By default the TV will warm up and cool dow
TVS '99
public beta
MUTING

1. Install the package appropriate for your distribution. For example, installing on Raspberry Pi OS would use the `.deb` package for `arm64`.

2. Once installed, you can run it from your application menu (under Audio / Video) or by running `tvs-launcher` from the terminal.

macOS



1. Open the downloaded DMG file and drag the `Television Simulator Launcher` app to your Applications folder.

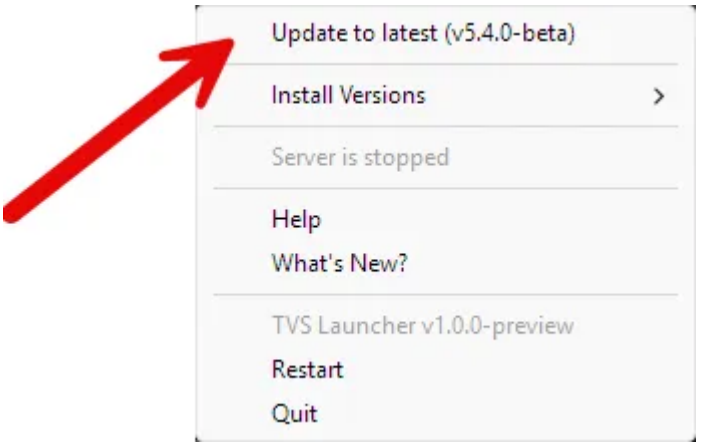
2. Run the app from your Applications folder. The Television Simulator Launcher icon will appear in your menu bar.

Usage

First run

1. Click the tray icon to open the launcher. When you first run it, no releases will be installed yet.

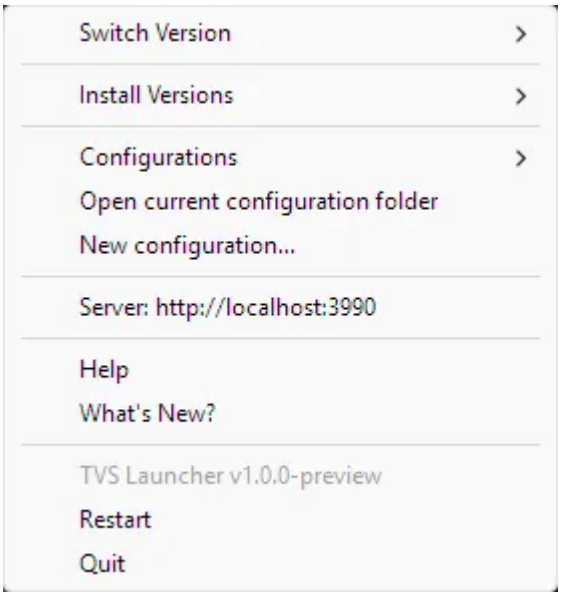
2. Click the **Update** button at the top to download and install the latest version of TVS.



Once updated, the server will start automatically, by default at `http://localhost:3990`

You can change the server port and host in the `settings.json` file located in the root folder.

Configuration Management



All of your user data and configurations are stored in the `configurations` folder. When updating to a new version, your existing data will be preserved. New configurations and installations will start with the latest demo config file.

Choose “Open current configuration folder” to open the folder where your current configuration is stored and edit your `config.tvs.yml` file, and the changes should be reflected when you refresh TVS in your browser.

Supporter Features

Supporters of Television Simulator (GitHub Sponsors, Ko-Fi, Discord, etc.) can access prerelease versions of TVS to help test new features and provide feedback before public release. To enable this, make sure you've created a supporter account (join the Discord and message me if you need help getting an account), then log in via your Gitea account username and password.

Using the CLI

The launcher also includes a command-line interface (CLI) that you can use to control it from a terminal or script. This is also useful for running the launcher on Linux servers without a graphical interface.

Run `tvs-launcher --help` to see the available commands:

> **Note**
>
> On Windows, you will need to run `tvs-launcher-cli.exe` to use the CLI.

Developer Portal

This section of the documentation is aimed at web developers who want to enhance their web applications to work seamlessly with Television Simulator.

We’ve just begun building out new developer features so stay tuned for more guides and API references in the future!

TVS IFrame API

Starting with Television Simulator 5.6.0, developers can access options from the TVS config file to use in their web applications. This allows TVS users to have an easy way to pass JSON data from TVS to your web application.

We use Window.postMessage() and IFrames to enable two-way communication between TVS and your application.

Handshake Process

Since: TVS 5.6.0

Your application is in control of when to start the handshake process with TVS; this is so you can ensure your app is fully loaded and ready before attempting to communicate with TVS.

To perform the handshake, your application should send a `TVS.Ready` message to the parent window (TVS) using the following code:

```
window.parent.postMessage({'type': 'TVS.Ready'}, '*');
```

Once TVS receives this message, it will respond with a `TVS.Options` message containing the configuration data.

```
window.addEventListener('message', (event) => {
    if (event.data.type === 'TVS.Options') {
        const options = JSON.parse(event.data.options);
        // Use the options as needed
    }
});
```

`event.data.options` will contain a JSON string of the IFrame options defined in the TVS config file.

Caution

To ensure consistency, TVS will always send the options even if no options were defined. In such cases, the options will be an empty JSON object: `{}`.

Options Format

Since: TVS 5.6.0

The `options` JSON string should resolve to a dictionary of key-value pairs. The keys will be strings.

The values can be of various types, including:

- Strings
- Numbers
- Booleans
- Arrays
- Nested objects

Note

TVS does not perform any type validation on the options; it's up to your application to handle and validate the data as needed.

Caution

Remember that **you** define the structure and types of the options in your application. Users will need to follow your documentation on how to set up the options correctly in the TVS config file.

Example Usage

Imagine you're building a weather application that can display information for different cities and you'd like to allow users to set their preferred locations via TVS. You might decide to accept a list of city names as an IFrame option.

Suppose you wanted to accept a list of cities like this:

```
{
    "cities": ["New York", "Los Angeles", "Chicago"]
}
```

You'll instruct your users to add this YAML to their TVS config file:

```
channels:
- number: 1
  name: "My Cool Weather App"
  iframe:
    src: "https://mycoolweatherapp.example.com"
    options:
      cities:
        - "New York"
        - "Los Angeles"
        - "Chicago"
```

Upon handshaking, your application will receive an event like this:

```
{
    type: 'TVS.Options',
    options: '"cities":["New York","Los Angeles","Chicago"]'
}
```

Integration with TV State

Since: TVS 5.7.0

In addition to `TVS.Options`, your application can also listen for other status messages from TVS:

- `TVS.Volume`: Reports the current volume level (from 0 to 100).
- `TVS.Mute`: Indicates whether the TV is muted.

- `TVS.Power`: Indicates whether the TV is on or off.

These messages will be sent automatically during the initial handshake and whenever the state changes in TVS.

Example that listens for these messages: 🔗

```
if (window.parent && window.parent !== window) {
    // Only attempt to handshake with TVS if we're in an IFrame.
    // Set up an event listener before attempting a handshake so that you're ready to receive the reply.
    addEventListener('message', function(event) {
        // These are the expected events from TVS when sending `TVS.Ready`

        switch(event?.data?.type) {
            case 'TVS.Options':
                console.log('Received options string:', event.data.options);
                const parsedOptions = JSON.parse(event.data.options);
                console.log('Parsed options object:', parsedOptions);
                // Use the options as needed
                break;

            case 'TVS.Volume': // Sent when volume changes and during handshake
                const volume = parseFloat(event.data.volume);
                // Update your app's volume accordingly; event.data.volume is between 0 and 100
                break;

            case 'TVS.Muting': // Sent when mute state changes and during handshake
                if (typeof event.data.isMuting === 'boolean') {
                    const isMuting = event.data.isMuting;
                    // Update your app's mute state accordingly
                }
                break;

            case 'TVS.Power': // Sent when power state changes and during handshake
                if (typeof event.data.isOn === 'boolean') {
                    const isOn = event.data.isOn;
                    // Update your app's power state accordingly
                }
                break;

            // As the API evolves, more message types may be added here.

            default:
                // Ignore other messages
                break;
```

```
        }
    });

    // To attempt the handshake, post `TVS.Ready` event:
    window.parent.postMessage({'type': 'TVS.Ready'}, '*');
}
```

Security Considerations

When implementing the IFrame API, it's crucial to consider security implications:

- Always validate and sanitize the data received from TVS to prevent potential security vulnerabilities.
- Be cautious when using the wildcard `'*'` in the `postMessage` method; consider specifying the exact origin of TVS if known. (If you are hosting TVS yourself and know where it will be accessed from, you can specify that URL instead of `'*'`. If you'd like anyone to be able to use your app with any TVS instance, you can leave it as `'*'`.)
- Implement proper error handling to manage unexpected messages or data formats.

Test Application

To help you get started with the TVS IFrame API, we've created a simple test application that demonstrates how to use the API to receive options from TVS.

It's a single-file HTML file to make it easy to read. View the source code to see how it works and use it as a reference for building your own applications.

Credits, Thanks, Acknowledgements

Television Simulator uses many 3rd-party resources which made it way better. Thanks!! These resources are owned by their creators and are not distributed under the same license as the code of the project as a whole.

3rd Party Software

3rd party resources included in Television Simulator like images, sounds and libraries are provided under different licenses specified by their respective authors. Here are a few:

- The Svelte framework and the many `npm` packages I used. Visit `package.json` to learn about these!
- PeerJS server for remote control functionality. Very cool WebRTC library and service!
- Scanlines CSS by Mehdi
- VCR OSD Mono Font by Riciery Leal
- New Prevue Fonts by rudyvalencia
- Home Video Font by GGBotNet
- Prevue reference software, information, and Curday.dat by AriX and other Prevue Guide forum members.
- Generic Mobile System Fonts by Jayvee Enaguas
- The awesome Butterchurn web GL audio visualizer
- The Weather Channel Icons set by TWC Classics contributor Nick Smith.
- WeatherStar 4000 Fonts set by TWC Classics contributor Nick Smith.
- Texscan MSI SG3 Font by Joe Strosnider
- Nostalgist.js video game emulator library

For a full list of every package used in this project, go to `/help/about` in the program or press `AltV` `AltV` to view version information for your specific version.

3rd Party Music and Artwork

On the project's official homepage and in example configurations, the following content is used:

- Vaporwave radio from Nightwave Plaza as background audio on some channels. Very cool service!
- Embedded YouTube videos from various creators. Search `youtube` in the `static` directory to find them all.
- Blender Open Movies as example video content.
- Placeholders from Greenhill County Public Works in the `static/placeholders` directory.

All content is either licensed under a license that allows it to be used, or is used with permission. However, if you previously gave me permission to use your content and would like it removed from the default config files, please contact me and I will gladly do so.

Special Thanks

Thanks to the following members of the community for helping with the project!

- sirocyl has provided some very believable CSS for TV transitions
- ChrisKewl has helped with Raspberry Pi 4 and 5 deployments and has tested many new features
- Electronic Search 3.0 has provided many high quality reference images and information about multiple satellite receivers
- BlockHead728 has tested on lower-end PC hardware to get a list of minimum and recommended PC system requirements

Support the Project!

Television Simulator is free to download and use! If you enjoy using it, please consider contributing to the project. Contributions don’t necessarily mean money; you can tell your friends about the project, report bugs, suggest features, or assist new users on the Discord or Slack servers.

If you use Television Simulator in a cool new way or on interesting new hardware, please share how it’s working out for you! I’d love to hear about it.

If you’d like to make a donation, you can contribute through Ko-fi, GitHub Sponsors, or if you’d prefer a shiny badge on Discord you can join there as a supporter. Every bit of support helps keep the project alive and growing! Funds will be spent on server costs, development tools, hardware for testing and coffee.

Thank you for your support!

Support via GitHub Sponsors →

Join the Discord and get a Supporter Badge →

Support via Ko-fi ☕ →

Supporter Benefits

Supporters can access prerelease versions of Television Simulator and the upcoming TVS Studio companion server. Join the Discord for more information. If you’re a supporter and have any questions or can’t get access, message me directly and I’ll get you set up as soon as possible!